Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Slicepedia
Open Corpus Slicing for Adaptive Web Systems

Killian Levacher

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

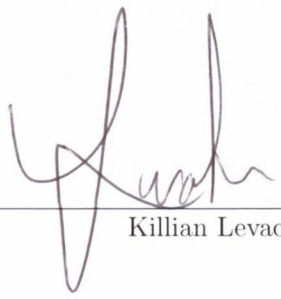
Doctor of Philosophy

October 2014

To my Parents

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work. I agree that Trinity College Library may lend or copy this thesis upon request.

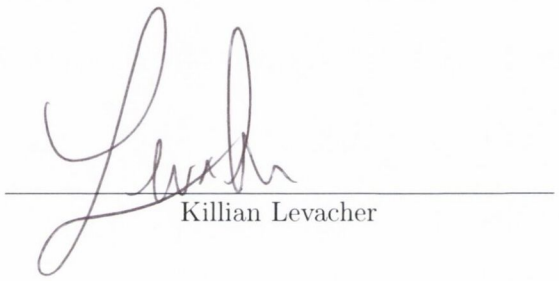


Killian Levacher

Dated: October 3, 2014

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.



Killian Levacher

Dated: October 3, 2014

Acknowledgements

Since setting sail on this solitary journey, I must admit a few existentialist thoughts, full of doubts and uncertainty, did cross my mind many times. Similar thoughts, which I imagine, Columbus himself must have pondered upon, on his first voyage to the new world. This solitary journey has now reached an end, and it does appear like a new shore is in sight after all. This journey couldn't have been possible, without the support of so many people, which I will always be grateful to.

I wish, first and foremost, to deeply thank my supervisor Prof. Vincent Wade, whose vast knowledge and experience, friendship, guidance and encouragements made this work possible. Special thanks are also reserved for Prof. Seamus Lawless, my co-supervisor during this research, whose continuous kindness, insightful advice and support throughout the years was invaluable.

I would also like to extend my gratitude to all of my family, and in particular to my parents, who have always believed in me, provided so much dedication to their children's education and to whom I will always be grateful to. My partner Claudia, for her never-ending support and whose love and incredible patience helped me persevere during many difficult times.

I would additionally like to convey my appreciation to all members of the Knowledge and Data Engineering Group in Trinity College Dublin, who's constant cheerfulness provided a fun and intellectually stimulating work environment in which to pursue this research. Finally, none of this work would have occurred, without the financial support of Science Foundation Ireland, to which I would like to express my gratitude.

Go raibh cead mile maith agat.

Ever tried. Ever failed. No matter. Try Again. Fail again. Fail better.

Samuel Beckett (1906 - 1989)

Ancora Imparo

Michelangelo (1475 - 1564)

Killian Levacher

University of Dublin, Trinity College

October 2013

Abstract

The growing demand for user experiences on the web addressing individual needs, is driving the mainstream adoption of personalisation technologies across broad fields of interests. Adaptive Web Systems (AWSs) have traditionally attempted to deliver dynamically adapted and personalised experiences to users, through the sequencing and adaptation of recomposable content. The adaptivity offered by these systems however, can be restricted by a lack of sufficient resources, which represents a major obstacle to their widespread adoption. So as to serve the largest possible audience, common sense encouraged the labor intensive and limited production of such resources, to target the common and predictable needs of users in higher demand, by delivering one-size-fits-all recomposable material, of type and quantity predetermined in advance. However, as the number of internet users grows, so does the diversity of user information needs. As a result, the need for even greater volumes of resources, covering larger ranges of increasingly unpredictable and specific user needs, rises too. Existing approaches incorporating open web corpus content within these platforms have so far achieved very limited success, mostly due to their inability to supply resources in a form meeting specific content requirements of AWSs. Within this context, the ability to supply adequate content to AWSs could only be sustained if such resources were produced in large volumes, only as needs arose. Such a content delivery service would remove the necessity for any content type/quantity predictions to occur, which in turn would enable the delivery of more scalable personalisation experiences, covering a wider range of needs.

The primary contribution of this thesis therefore consists of an innovative approach to open web corpus reuse and incorporation within AWSs, named Slicepedia. Through the identification and development of dynamic means of open web content harvesting, customisation and delivery, a fully-automated tool-chain service can be made, which delivers recomposable open web corpus resources, as right fitted packages to independent AWSs. This approach differs significantly from those used by existing AWSs, as it enables the on demand reuse of open web corpus content, without any pre-determined conditions and based upon a variety of possible AWS content requirement combinations, undetermined in advance. In order to evaluate this research, a series of user-trials and component analysis are presented, which provide confidence in the ability of this service to deliver open web corpus resources to AWSs as customised packages, with a suitability similar to content purposely produced for such systems and with minimal loss in original quality. The analysis also provides confidence that the on demand customisation, delivery and reuse of open web corpus resources within AWSs is possible and that such a service could support the widespread adoption of more scalable AWSs.

Contents

List of Figures

[1] http://lod-cloud.net/ [Accessed: October 3, 2014]

[2]From webpage: http://www.bbc.co.uk/news/world-mi ddle-east-24068867

List of Tables

Acronyms

AAF Automated Annotation Focused Content Batch.

ACME Align, Collapse under Mismatch, and Extract.

AHS Adaptive Hypermedia System.

AMAS Adaptive Media and Services for Dynamic Personalisation and Contextualisation.

ANNIE A Nearly-New Information Extraction System.

APeLS Adaptive Personalized eLearning Service.

API Application Programming Interface.

ARCHING Adaptive Retrieval and Composition of Heterogenous INformation sources for personalised hypertext Generation.

ARI Adjusted Random Index.

ATF Automated Topic Focused Content Batch.

AWS Adaptive Web System.

BESUS Bengal Engineering and Science University, Shibpur (India).

BO Basic Object.

CAB Content-Agnostic Batch.

CAS Content Adaptation Spectrum.

CCR Content to Code Ratio.

CNGL Centre for Next Generation Localisation.

CPU Central Processing Unit.

CRF Conditional Random Field.

CSB Content-Specific Batch.

CSS Cascading Style Sheets.

DARPA Defense Advanced Research Projects Agency.

DCF Densitometric Content Fragmentation.

DI Design Influence.

DOM Document Object Model.

DR Design Requirement.

EAHS Educational Adaptive Hypermedia System.

GAE Google App Engine.

GATE General Architecture for Text Engineering.

GRAPPLE Generic Responsive Adaptive Personalized Learning Environment.

HMM Hidden Markov Model.

HTML HyperText Markup Language.

HTTP HyperText Transfer Protocol.

IE Information Extraction.

IR Information Retrieval.

JAPE Java Annotation Patterns Engine.

JWPL Java Wikipedia Library.

KP Key Principle.

LOM Learning Object Model.

LOR Learning Object Repository.

LR Language Resource.

MAF Manual Annotation Focused Content Batch.

MCB Manual Content Batch.

MERLOT Multimedia Educational Resource for Learning and Online Teaching.

ML Machine Learning.

MTF Manual Topic Focused Content Batch.

NCB Native Content Batch.

NDLR National Digital Learning Resources.

NER Named Entity Recognition.

NLP Natural Language Processing.

OAHS Open Adaptive Hypermedia System.

OCCS Open Corpus Content Service.

OCR Optical Character Recognition.

ODP Open Directory Project.

OOPS Ontology-based Open-corpus Personalization Service.

PDF Portable Device Format.

PDoC Permitted Degree of Coherence.

PIR Personalised Information Retrieval.

PMCC Personal Multilingual Customer Care.

POJO Plain Old Java Object.

POS Part-Of-Speech.

PR Processing Resource.

RAM Random Access Memory.

RDF Resource Description Framework.

REST REpresentational State Transfer.

SaaS Software as a Service.

SCI Self-Contained and Independent.

SCO Sharable Content Object.

SCORM Sharable Content Object Reference Model.

SFI Science Foundation Ireland.

SPARQL SPARQL Protocol and RDF Query Language.

SQL Structured Query Language.

SR Slicepedia Request.

SRC Slicepedia Requests Coordinator.

SST Site Style Tree.

TCP Transmission Control Protocol.

TDR Technical Design Requirement.

TEL Technology Enhanced Learning.

TTR Text-to-Tag Ratio.

UML Unified Modeling Language.

UNITE UNIfied Task-based browsing Experience.

URI Universal Resource Identifier.

URL Universal Resource Locator.

VIPS VIsion-based Page Segmentation.

VM Virtual Machine.

WBT Web Based Training.

WCB Wikipedia Content Batch.

WWW World Wide Web.

XML Extensible Markup Language.

Chapter 1

Introduction

1.1 Motivation

The ubiquitous and instantaneous access to information offered by the internet is fundamentally changing how users expect information to be presented and delivered to them. Until recently, digital access to information on the World Wide Web (WWW) was mass-oriented [Kobsa2001]. Decisions regarding content production to support information services were geared to suit the largest possible audience. Common sense encouraged the *labor intensive* and *limited production* of resources to target the *common* and *predictable needs* of users in higher demand, by delivering *one-size-fits-all* material. However, as the number of internet users grows, so does the diversity of user information needs. Since 2006, the number of internet users has more than doubled, reaching a total of 2.8 billion users, with an increasing number of individuals accessing the web through mobile devices and from more diverse backgrounds (different languages, cultures etc) [ITU2013].

As a result, the growing demand for more online user-experiences addressing increasingly smaller niche needs [Anderson2006] is driving the mainstream adoption of personalisation technologies across broad fields of interests. Adaptive Web Systems, also commonly referred to within the research community as Adaptive Hypermedia Systems (AHSs)[1] have traditionally attempted to deliver dynamically adapted

[1]While the vast majority of Adaptive Hypermedia Systems are today constructed as Adaptive Web Systems, the work presented within this thesis could be applied to other forms of hypermedia.

and personalised experiences to users on the web, through the sequencing and adaptation of recomposable resources. While the effectiveness of such systems and the benefits of their use have been proven in numerous studies [Conlan2004a, Plate2006, Steichen2009a, Brusilovsky1998a, Hook1997b], the adaptivity offered by AHSs can be restricted by a lack of sufficient resources in terms of *volume* and *content requirements* (granularity, style and meta-data), as well as *interoperability* between AHSs [Brusilovsky2007h]. Such adaptive systems have traditionally relied upon the manual and labor-intense production [Dieberger2002] of bespoke proprietary closed corpus content[2]. The result, typically is the production of relatively low volumes of recomposable resources (produced in most cases at high costs), which represents a major obstacle to AHS widespread adoption [Brusilovsky2007h, Conlan2002].

Furthermore, as the necessity to target smaller niches increases, so does the need for even greater volumes of resources, covering larger ranges of information needs. As pointed out by [Steichen2011a], "the production of [AHS] resources a-priori of any [user] interaction, generally assumes that the *type* and *quantity* of resources needed for a particular AHSs is known in advance of system deployment". However, in a scenario involving ever larger numbers of users, with more specific and diverse needs, the range of possible content type requirement combinations becomes very large and hence the number of requests also spreads across this range. As a consequence, the difficulty involved in predicting the type and quantity of content to be produced in advance naturally also rises.

Within this context, the ability to supply adequate resources needed by AHSs (to provide personalisation on such a large range of user needs) could only be sustained if these resources were *produced only as needs arose*, which would remove the necessity for any content type/quantity predictions to occur. Such a novel and hypothetical content production service would be required to produce resources i) in large volumes, ii) on demand, iii) automatically and iv) at low costs, v) suitable for reuse within various AHSs to support an undetermined range of activities.

For this reason, the broader and more conventional term hypermedia is used throughout this thesis as opposed to the latter.

[2]The definition of what constitutes closed corpus resources, is formalised in section 2.3.3

Open Adaptive Hypermedia System (OAHS) research addresses parts of this challenge by leveraging open corpus resources[3] available on the WWW and utilising them within AHSs. This field of research aims to provide AHSs with a set of resources covering a more open range of information needs. However, when open corpus reuse has been achieved, it is traditionally performed *manually* [Henze2001a] or at best using automated approaches that treat such resources as document level packages only (section 2.3.4). The usage of these techniques has met with very limited success, even when incorporating relevant open web information. These content supply techniques suffer because they only provide *one-size-fits-all, un-customised* delivery of results, with limited control over the granularity, content format or associated meta-data of resources targeted for reuse by AHSs [Levacher2012d]. Open web corpus material, in its native form, is very heterogeneous. It comes in various formats, languages, is generally very coarse-grained and contains unnecessary noise such as navigation bars, advertisements etc.[Weissig2009, Gottron2008a]. As pointed out by [Lawless2009], *there is an inverse relationship between the potential reusability of [...] content and its granularity.* The reuse of one-size-fits-all open web resources could be improved if reused in different sizes and removed of superfluous content such as navigation bars, advertisement etc.

Furthermore, in order to be reused for purposes not necessarily planned by the original open corpus content authors, AHSs require the resources supplied to meet specific content requirements. In other words, in the same way copper or gold are mined as raw materials, processed and further reused for various purposes, open web corpus resources should be harvested as preliminary resources and converted into reusable *tailored* content packages, serving specific content requirements needs of individual AHSs. A one-size-fits-all delivery of open web corpus content to AHSs would be equivalent to the reuse of raw copper material without any post-excavation modifications.

More recent OAHS approaches [Steichen2012b, Weal2007, Tang2010] have attempted to semi-automatically tailor open web content through the use of hand-crafted

[3]The definition of what constitutes open corpus resources is formalised in section 2.3.4

resource specific algorithms, for the purpose of reuse within predefined AHSs. Such approaches to open corpus reuse are referred throughout this document as content/AHS specific techniques[4]. Although these techniques provide higher control over the granularity and reuse of such resources, only a *pre-defined set of open web resources* can be targeted for reuse, for *pre-determined AHSs, content requirements and reuse cases.* Whenever new resources or different niche content requirements[5] emerge, existing algorithms must be changed, or replaced altogether. These techniques, in essence, attempt to deal with the challenge of open web corpus heterogeneity by reducing the degree of unpredictability and variability inherent to such resources and AHS content needs, which limits the scale of reuse.

The key motivation of this research is to improve the reuse of open web corpus resources within independent[6] AHSs, by reducing the impact caused by both the unpredictability of resources targeted for reuse or intended reuse purposes.

Through the identification and development of dynamic means of open web content harvesting, customisation and delivery, a service can be made which meets the requirements of the hypothetical content production service outlined above. More specifically, a fully-automated (iii) service can be made which provides the on-demand (ii) supply of large volumes of recomposable resources (i), meeting arbitrary AHS-specific niche content requirements (v), at low cost (iv).

The dynamic reuse of open corpus resources available on the web can be performed without the need for any open corpus or AHS content requirements pre-conditions to be met. Such an approach is referred throughout this document as content/AHS agnostic open corpus reuse[7]. Such a service would enable the delivery of more scalable personalisation experiences, covering a wider range of needs, and would empower AHS designers to concentrate their decisions upon optimal system adaptation behaviour rather than content production.

[4]The definition for content/AHS specific techniques is formalised in section 2.3.4.4

[5]The definition of niche content requirements is formalised in section *b*) and the glossary

[6]The definition of independent AHSs is formalised in the glossary

[7]The definition for content/AHS agnostic techniques is formalised in the glossary

1.2 Research Question

The research question investigated in this thesis is: *What are the techniques and technologies required to improve the reuse of open corpus resources within independent AHSs?*

By reuse improvement, this thesis focuses upon the ability of AHSs to request open-corpus resources *i*) without any pre-determined conditions *ii*) as right-fitted content packages, *iii*) based upon a *variety* of possible AHS content requirement specifications, undetermined in advance[8].

This research is focused upon the design and development of a service improving the reuse of open corpus content available on the WWW. The aim of this service is to enable the harvesting, right-fitting[9] and delivery of open web content as part of customised content packages, which can be directly incorporated and recomposed within AHSs. The reuse of the content packages, which are generated, should focus upon the adequate integration across various independent AHSs as opposed to predefined and dedicated content consumers.

1.3 Research Goals and Objectives

The proposed research question, can be subdivided into three major objectives namely:

1. To investigate and specify techniques and technologies which can be used to enable automated reuse improvement of open corpus resources capable of supporting niche content requirements of independent AHSs.

2. To design and implement a content/AHS agnostic slicing[10] service, based upon

[8]This definition of reuse improvement is established in particular in opposition to traditional Information Retrieval (IR) services, which currently only provide the ability to request open-corpus resources as one-size-fits-all packages, in an arbitrary format, based only upon content requirement specifications limited to keyword-based queries (section 2.3.5)

[9]A formal definition of right-fitting is available in the glossary

[10]The concepts of both slicing in general and content/AHS slicing are formalised in section 3.3.2

these techniques, which can provide the on-demand harvesting, customisation and delivery of open corpus content to AHSs.

3. To evaluate the extent to which such a service can improve the reuse of open corpus resources within independent AHSs

1.4 Research Approach

As presented in the previous section, the main contribution brought forward by this research focuses on the investigation of a novel reuse mechanism for open corpus resources. In particular, this research focuses upon the reuse of such resources as part of a tool chain, combining different fields of research together in order to improve their reuse as part of AHSs.

Since investigating in detail each individual fields of research, combined for the purpose of this tool chain, would have been impractical, the research process undertaken for this investigation aimed instead to evaluate as a priority the feasibility and limits of such reuse as an overall approach. Assuming, the investigation carried out validates this approach, individual fields of research corresponding to each component could be improved separately, in future research, to further enhance the performance of the tool chain.

The rest of this section presents the overall steps carried out as part of this research.

In order to accomplish the research goals and objectives identified above, it was first necessary to conduct a review of content reuse approaches currently in use within the research community. Chapter 2 of this thesis hence, presents a state of the art analysis of content supply and reuse methods used in general and more specifically within the AHS community. An identification of opportunities and limitations of current approaches is presented, followed by a review of content analysis and adaptation algorithms. During the process of carrying out this review, it became increasingly clear that the ability to fragment open corpus resources would represent a critical component of a reuse tool chain. For this reason, an important section of this chapter is dedicated to such algorithms.

The vision of a novel content supply service, supporting the automated reuse and customisation of open web resources for consumption within AHSs, is then outlined in chapter 3. Influences resulting from state-of-the-art investigations, presented in the preceding chapter, are summarised leading to the selection of specific design requirements which this service should support.

A technical description of Slicepedia, a novel tool chain application automating the customised reuse of open web content is then provided in chapter 4. Slicepedia is a service which provides AHSs with customised and recomposable content packages, automatically produced on demand, through the automated reuse of open corpus resources. An overall description of the system's architecture is presented along with technologies used to implement the design requirements enunciated in the preceding chapter.

This thesis subsequently proceeds with a description of the various analysis and user trials carried out for the purpose this research. Each evaluation is aligned with the research question and objectives enunciated in section 1.2. The overall evaluation of this research ranged from quantitative component-specific measurements up to comprehensive system-wide user based trials. Trials were conducted across various content reuse vehicles for which individual technical characteristics relevant for each experiment are explained. Results obtained for each experiment are then presented in relation to corresponding initial research objectives. More specifically, the first experiment presented in this chapter aimed at evaluating the performance of a fragmenter algorithm selected from the review carried out in chapter 2. Since the adequacy of this algorithm for the purpose of this tool chain was unknown at the time this research was carried out, the hypothesis of this experiment focused upon validating this fragmenter selection as well as identifying its performance with respect to additional characteristics missing at the time in the literature.

While the first experiment focused upon one critical component of the tool chain, the last two experiments on the other hand, aimed at investigating directly the overall reuse approach proposed by this research. The hypotheses of these experiments aimed initially at evaluating whether a slicing approach, in ideal condition and with

a dedicated AHS, does improve the reuse of open corpus resources or not. Assuming it does, the subsequent hypothesis thereafter investigates the tradeoffs involved when using a content-agnostic approach to slicing. The hypothesis in the last experiment, subsequently focussed on investigating whether such an approach can represent a valid solution to supply open corpus resources for recomposition as part of various independent AHSs.

While investigating the fragmenter algorithm selected for this research, it became apparent that the performance of this fragmentation approach was dependent upon the type of open corpus resources targeted. Although, this clearly represents a drawback which will need to be addressed in the future, since the diversity and volume of resource targetable by this algorithm is still very large, this disadvantage did not impeded the goal sought by this research with respect to validating the overall reuse approach. The last two experiments presented in this chapter where therefore carried out upon this subset of open corpus resources.

This thesis finally concludes with a description of the objectives and achievements carried out within this research. The key contributions provided to the state of the art of open content reuse are summarised and presented in chapter 6. Finally, a discussion of appropriate future work, which unfolds based on the outcomes of this research is set forth.

Although there are several research fields which intersect with this investigation, certain areas are deemed outside the scope of this thesis. As the challenge of identifying web resources relevant to a particular query is a very well documented and established field of research of its own (section 2.3.5), this research considers out of scope this issue and instead focuses upon the customisation and delivery of such resources for reuse within AHSs. Although the use of an independent content supply system for AHSs unveils new opportunities with respect to the range and specificity of personalisation experiences possible, it is not within the scope of this research to deliver next generation AHSs as such (section 6.3). The exploration of resulting repercussions upon possible new application use cases will need to be addressed by future research. Finally, in order to deal with the significant technical challenges of

open corpus content right sizing and reuse, copyright and digital rights, security and privacy management issues will also be considered out of scope of this research.

Chapter 2

State of the Art

2.1 Introduction

This chapter provides the reader with an insight into AHSs, as well as content reuse and analysis techniques in general. The aim is to provide the reader with an overall context of the research area, so as to extract relevant limitations and influences from the state of the art, which will subsequently be used to develop design requirements in the following chapter. The aim is also to contribute to the first objective of this research by presenting a set of technologies and approaches, which could improve the reuse of open corpus resources within AHSs in general.

Section 2.2 of this chapter therefore initially presents the reader with a general overview of AHSs, their anatomy and most common hypermedia adaptation techniques used, for the purpose of delivering personalised experiences to individual users. This section solely intends to provide a brief overview of these systems and techniques for the purposes of design requirement extraction in the next chapter.

Following this overview of AHSs, section 2.3 concentrates upon describing and analysing the various content supply models used by these systems. As most of the research presented in this thesis deals with this aspect of AHSs, this area is covered in more details than the previous section. The aim is to provide a context for further discussion related to limits of open corpus incorporation techniques (section 2.3.4).

Section 2.4.2 and 2.4 subsequently presents approaches and technologies used to reuse content in general and opportunities available in fields of research dedicated to the analysis of resources in general. At an early stage of this research, the component responsible for the fragmentation of open corpus resources was identified as a critical element of any slicer implementation. At that time, this field of research appeared to be still at an early stage, with no clear overall consensus as to best practices or technologies to use. A review of these techniques was therefore performed. This review, which represents an important part of resource analysis techniques presented in this chapter, served as a basis for the selection of a fragmentation approach to be used for in chapter 4.

2.2 Overview of AHSs

2.2.1 Introduction

In a society were swift access to information is becoming critical, AHSs are becoming increasingly popular tools for user-driven access to information [Brusilovsky1996a]. AHS is an area of research which combines the fields of hypertext and user modelling. Given a diverse user population, a web or traditional hypermedia system will suffer from the inability to be "all things to all people" [Brusilovsky2001]. AHSs offer an alternative to the traditional one-size-fits-all hypermedia experience, by adapting various visible aspects of the system to the goals, learning style, and knowledge [Conlan2002, DeBra2006a, Brusilovsky1996] of individual users as they interact with the system [Henze2002a].

Since Brusilovsky introduced the first classification of AHSs [Brusilovsky1996a], the field of adaptive hypermedia research has grown rapidly and has resulted in the development of many approaches and systems. This section therefore does not intend to provide an exhaustive list of AHSs, architectures or adaptation approaches, but instead to present the most influential systems and techniques, used within this field as a basis for further discussion in subsequent chapters of this thesis. A more detailed description of these systems, can be found in reviews carried out by [Knutov2009] or [Brusilovsky2001].

2.2.2 Anatomy of AHSs

Although a variety of AHSs have been built throughout the years, there is still no consensus as to what is the ideal architecture and anatomy of such systems [Knutov2009]. To this day, new models [Knutov2010], which attempt to unify modern approaches, are still being proposed in order to take into account latest adaptive aspects and practices introduced by the community. A first attempt to provide a generic architecture for AHSs was proposed by [DeBra1999f] with the AHAM model. Since this model possesses most of the components currently encountered in modern AHSs, it represents a good reference point for subsequent AHS architectures presented

in this document. As can be seen in figure 2.1, in order to provide a personalised experience, AHS designers are required to specify how the information presented to individual users will be adapted based upon changes which occur within a domain and user model. This behaviour is generally encoded in the form of rules, triggered while the user interacts with the platform. It produces a detailed specification, describing how the adaptive presentation should be generated for a given user. This set of rules is generally referred to as an adaptation model and is interpreted by an adaptation engine. A runtime component thereafter consumes this specification and builds the presentation displayed to individual users, based upon what content was selected and how it is composed together. The domain model describes how information or knowledge modelled within an AHS is structured. It usually represents this information as a set of concept hierarchies, along with their relationships and mutual dependencies. The user model on the other hand, represents arbitrary characteristics of users taken into account by a AHS such as current knowledge, goals, learning style etc. Knowledge within a user model usually refers to concepts provided by the domain model and can be updated by rules specified within the adaptive engine. [Dimitrova2003] for example, proposes the concept of an open user model (open to the user for inspection), constructed by importing existing concepts from a domain ontology [Bontcheva2004] and updated through the interaction of the user itself with the system. Finally, the within-component layer in the AHAM model refers to the content available to the AHS for presentation purposes. This layer, also referred to as the content model, represents both what resources are available to the system and how they are connected to each other. Each resource is addressable through the anchoring layer which enables a separation of concern between how the content is stored and how it is being used by the adaptation engine.

Over time, AHS architectures have evolved increasingly towards more modular and component based architectures [Brusilovsky2001], each in turn attempting to apply and improve separation of concerns between preeminent functions of systems developed. As can be seen in figure 2.2, the same principles and basic components are employed by the Adaptive Personalized eLearning Service (APeLS) model introduced by [Conlan2002] a few years later. Prior to APeLS however, the pedagogical

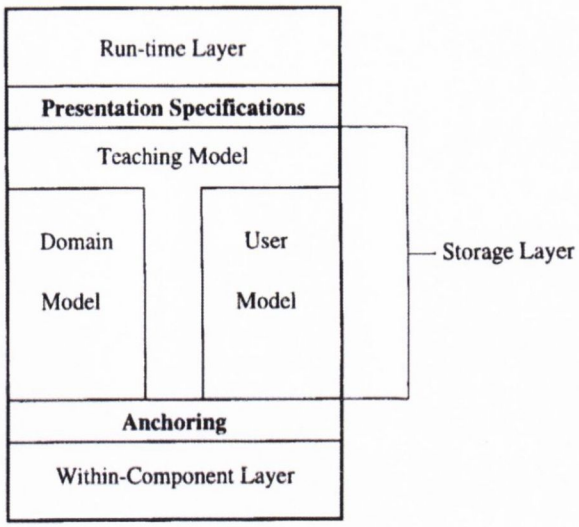


Figure 2.1: AHAM Model [DeBra1999f]

models (case based learning, simulations etc.) used in educational AHSs were either embedded in the content or into the adaptive engine itself. This situation required the re-authoring of the content or adaptive models whenever a new pedagogical model was introduced. For this reason, Conlan proposed in APeLS, an additional separation of concern regarding the pedagogical aspects of AHSs encapsulated within a narrative model. This multi-modal metadata-driven model encodes a set of generic strategies for presenting concepts, based upon various pedagogical approaches. Selected narratives and content are thereafter consolidated within the adaptive engine, which facilitates the reuse of learning resources across different pedagogical models. More recent research has brought this separation of concern and AHSs modularity yet a step further, through the distribution of various AHS functionalities across the internet via web services [OKeeffe2012, Henze2005, Staikopoulos2012]. [Steichen2011a] for instance, shows how even concerns related to content models can be delegated to an external web service.

2.2.3 AHS Adaptation Techniques

Once an AHS possesses the necessary data within its various models, it is ready to perform various **adaptation techniques** to deliver a personalised experience to each individual user. Although many classifications are possible, the following

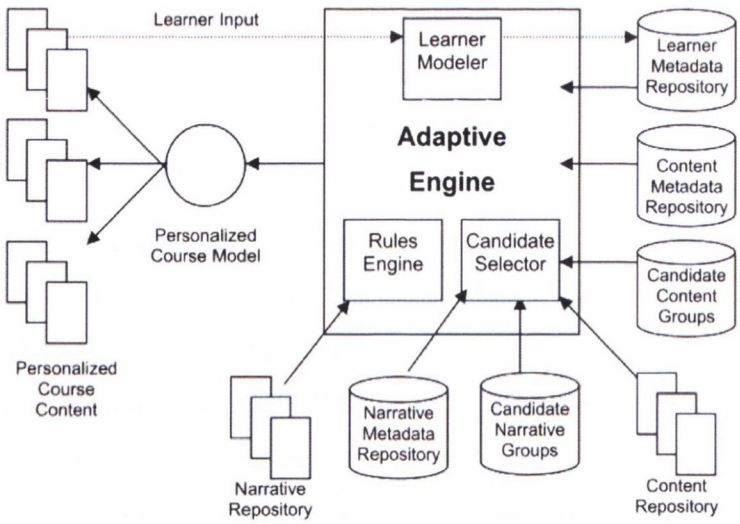


Figure 2.2: APeLS Model [Conlan2002]

taxonomy[1] was selected based upon the resource supply and/or reuse implications involved within AHSs, which will become relevant for the rest of this document. From the perspective of AHS resources these techniques can hence be classified into three separate categories namely, i) resource modification, ii) resource sequencing and iii) resource recomposition techniques.

Resource modification techniques aim to modify resources presented to individual users based upon their goal, knowledge or preferences. A large number of these modifications make use of link adaptation techniques or relevance emphasis techniques. Link adaptation techniques aim to improve the overall browsing experience of individual users by either i) hiding [DeBra2006b, Brusilovsky1998a], ii) ordering [Kavcic2004, Smyth2003], iii) annotating [Koidl2011, Zellweger1998a] or iv) automatically generating links [Kavcic2004]. Systems such as the UNIfied Task-based browsing Experience (UNITE) [Koidl2011] service for example, uses link annotation techniques to provide cross-site personalisation to users. It does so by annotating links with circular icons (figure 2.3) filled with a colour based upon the topical relevancy of the targeted page with respect to a current web search session. Relevance emphasis techniques, on the other hand, aim instead to help individual

[1]Since the slicing of video and graphic medium is considered out of scope of this research (section 3.3.2), adaptation techniques specific to these mediums are also omitted

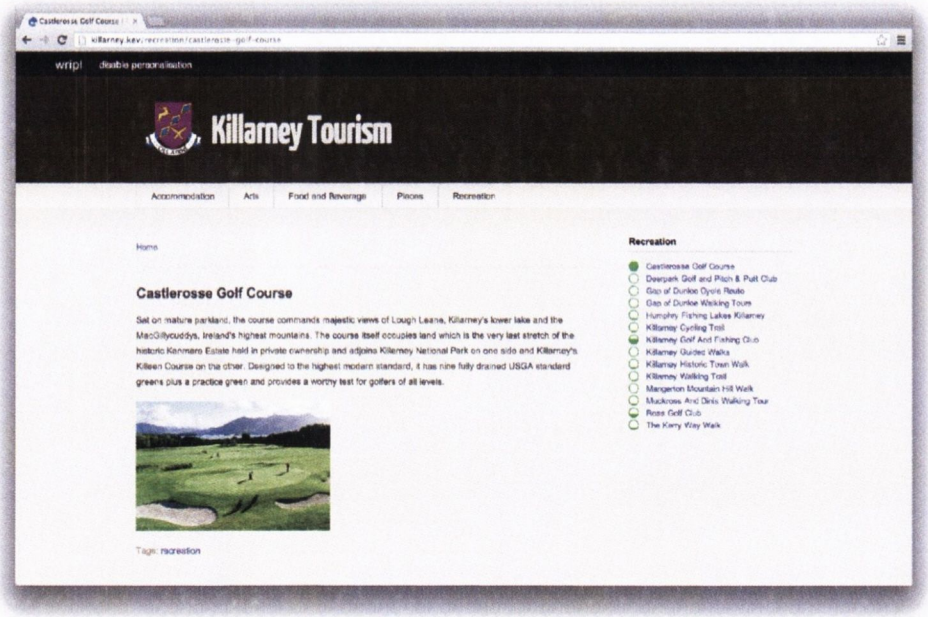


Figure 2.3: UNITE Screenshot [Koidl2011]

users focus their attention in priority upon the most relevant information presented to them, by using techniques such as i) stretchtext [Hook1997b, Weal2007], ii) sorting, iii) dimming [Brusilovsky1996a], iv) highlighting [ChenitiBelcadhi2006] and v) scaling [Tsandilas2004]. Stretchtext techniques for example, help users focus on relevant content by hiding contextual information of less relevance behind collapsible/hideable panes. Placeholders, usually containing a short header summarising the hidden information, signals to users the presence of additional content (figure 2.4).

While previous techniques focus upon adapting the way in which existing resources are presented to users, *resource sequencing* techniques on the other hand focus instead upon the decision mechanisms used to determine in which order (or at what moment) resources available to a AHS are sequenced (or suggested) to individual users. Sequencing decisions can be made based upon i) prior behaviour of a single or large group of users with a specific set of resources [Wexelblat1999] or ii) based upon various attributes of the content itself. Concept based methods [Staikopoulos2013, Brusilovsky2007b] for example, assign one or multiple concepts (typically contained within the domain model) to each resource, in order to determine which resources to

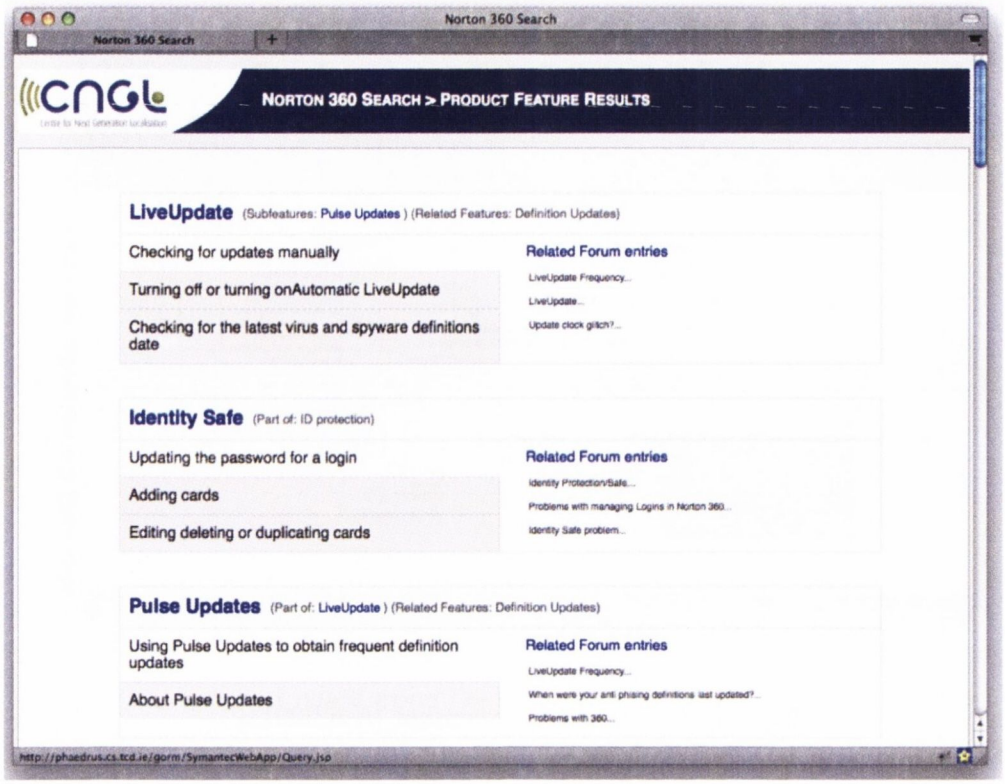


Figure 2.4: Stretchtext example [Steichen2011a]

present, based upon each individual user model.

Fragment variant approaches [Henze2001a], or *resource recomposition* techniques are also extensively used as adaptation mechanisms within AHSs. AHSs using this technique construct pages displayed to users in real-time, by selecting and recomposing appropriate self-contained fragments (text paragraph, pictures, videos etc) into coherent page-level resources. Final pages presented to users using this technique are only created once the exact combination of user needs has been determined by the AHS, which reduces significantly the level of labor involved when creating these resources. Steichen et. al [Steichen2011a] for example use this technique as part of the Personal Multilingual Customer Care (PMCC) system. As depicted in figure 2.5, multilingual pages presented to users are composed of individual monolingual fragments. The fragments are selected in realtime based upon both the combination of languages understood by each user as well as search results available in the system.

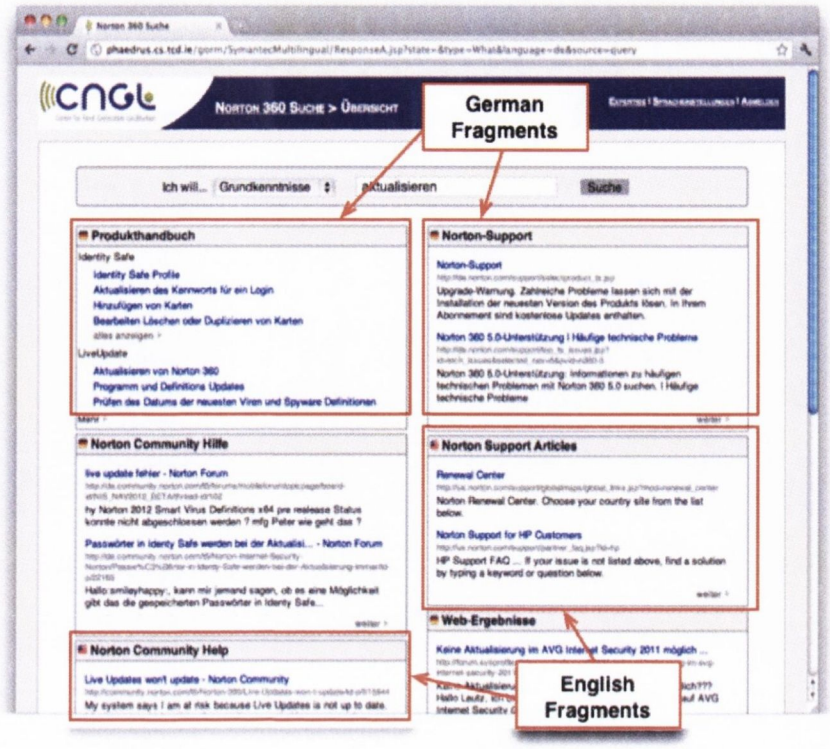


Figure 2.5: PMCC Recomposition Screenshot [Steichen2012b]

2.2.4 Conclusion

This section presented to the reader with an overall description of the anatomy and adaptation techniques used by AHSs in general. Since this research is primarily concerned with the content models and supply aspects of these systems, the following section of this document will concentrate in more details upon this aspect of AHSs in particular

2.3 AHS Content Models & Supply Paradigms

2.3.1 Introduction

The previous section of this chapter presented an overall view of the anatomy of AHSs along with the adaptation techniques they rely on, in order to deliver personalised experiences to individual users. This section in contrast, presents in more details aspects of AHSs related to content models. In particular, it initially describes the requirements such models must seek to fulfil, based upon adaptation techniques performed by AHSs, and subsequently describes the different techniques which are used to implement these requirements along with their limitations.

As the second objective of this research aims to develop an automated content supply system to be used by various independent AHSs, content supply models that rely upon manual or community driven approaches are presented for the purpose of clarity, however are not discussed in considerable details. Since this objective is to develop a content supply system independent of AHSs, this implies such a service does not have access to a dedicated AHS user base to support its content production and hence only represents a sideline of this research. Additionally, as the retrieval of open corpus resources through IR techniques is considered out of scope of this thesis (section 3.3.2), an overview of these techniques is presented to the reader as opposed to a detailed description. With respect to both of these cases, references to reviews, which focus specifically upon these fields, are provided to the reader.

2.3.2 Content Models Overview

In order to support an AHS, a content model must enable the delivery of resources which match various arbitrary combinations of requirements, provided by a given AHS. This set of requirements is referred to, within this thesis, as AHS content requirements[2]. AHS *niche content requirements* refer to one unique combination of resource attributes, which a particular AHS requires in order to provide an arbitrary

[2]The difference between what constitutes AHS niche and broad content requirements is formalised in the glossary

niche personalisation experience to an individual user. *Broad content requirements* on the other hand, refer to the collection of various niche content requirements required by one or many AHSs[3]. The set of content requirements produced by an AHS indirectly results from the personalisation experience intended to be produced for a given individual user. An AHS requires resources supplied to match these requirements in order to support the adaptation techniques presented in section 2.2.3.

In addition to creating resources in the first place, the production or conversion of a set of hypermedia resources in a form which is deliverable to AHSs, requires individual resources available within a given content model to i) be produced with properties matching the broad potential content requirements of an AHS, and ii) be augmented with additional information supporting its adaptive reuse. This additional information is referred to by [Brusilovsky2007h] as *adaptation-specific information.*

In order to guarantee an AHS will have sufficient adequate resources to recompose together and present to individual users, AHS content model designers must guarantee sufficient resources with adequate properties such as language, style (list, paragraph), media type (text, image, video) have been produced/included within the repository used as a basis for the content model. If fragment variant adaptation techniques are to be used by the AHS, content must also be created as self-contained resources of adequate granularities.

With respect to adaptation-specific information, these can be separated in four different subcategories consisting of: i) attribute, ii) inter-document, iii) external model connection and iv) intra-document information.

Attribute information simply consists of metadata describing the type and properties of documents available within the content model. Resources within the KBS-Hyperbook AHS [Henze2001a] for example can be marked of type "introductory", "quiz", "example", etc. The Adaptive Retrieval and Composition of Heterogenous INformation sources for personalised hypertext Generation (ARCHING) AHS also describes resources based upon properties such as "language", "size" etc.

[3]The definition of AHS content requirements is formalised in the glossary

This information is used in order to identify and retrieve the most adequate documents within the corpus, matching as close as possible the content requirements requested by a AHS.

Inter-document information on the other hand, assigns relationships as links between these resources in order to produce a hyperspace network between documents. Links can also be assigned attributes (as for documents). This information is used to enable user navigation between resources and also support link adaptation techniques (section 2.2.3) used by AHSs to guide users to the most appropriate documents.

Document *connections to external models* can also be added to provide additional knowledge behind the hyperspace. There exists a vast variety of models that can be used for this purpose such as conceptual, pedagogical or goal models for example [Brusilovsky2007h]. These connections are necessary to support content concept-based sequencing techniques (section 2.2.3) performed by AHSs. The most popular approach consists of connecting documents with a domain model. In the InterBook AHS for example [Brusilovsky1998c], documents are connected to a domain model with links typed based upon whether the domain concepts assigned to them are outcomes or prerequisites to this resource.

Finally, *intra-document information* providing details about internal characteristics of a document (topic covered by paragraph, title section of a resource) can also be added to resources. This information provides AHSs with the ability to provide resource modification adaptation to users by applying for example relevance emphasis techniques (section 2.2.3) on selected sections of resources as opposed to others.

The following sections describe various approaches which are used to produce such content supply models to support AHSs.

2.3.3 Closed Corpus Content Delivery Models

Early pioneer AHSs [DeBra1998, Brusilovsky1996, Dimitrova1998, Silva1998] aimed at exploring the potential benefits of hypermedia adaptivity and the possible techniques that could be used to achieve it (section 2.2.3). For this reason, AHS research was mostly focused upon what could be achieved by these systems, given the

adequate content was available to use, rather than the production of the content itself. Hence, content models of these early systems were based upon sets of resources, *manually handcrafted* specifically for the purpose of supporting the adaptation technique demonstrated by each individual AHS [DeBra1997]. Such content models are now referred to as closed corpus content models. A closed corpus of documents, consists of resources where documents, their attributes and relationships between documents and external models, are known to AHS designers at system design time[4]. A closed corpus AHS (referred in this thesis as traditional AHS) is therefore an adaptive system which operates on a closed corpus of documents.

What makes this content model so special is that resources available in these systems are highly curated and match AHS content requirements with high precision. Hence it enables AHSs to be supplied with the ideal resources needed to deliver chosen adaptation techniques effectively to users, which in turn allows the full power and benefits of adaptivity to be explored.

However, although the benefits of such systems have been demonstrated on numerous occasions (section 2.2), they have so far failed to become wide spread [Brusilovsky2007h]. The overall range of recompositions and adaptivity, delivered by an AHS to individual users, is ultimately restricted by the diversity and volume of resources put at its disposal [Levacher2011]. Because closed corpus content models are built manually, they do not scale and are impractical for most real world applications. No content provider is able to invest the necessary time to structure and index thousands of documents as required by these systems (section 2.3.2); worse, these systems need constant maintenance, as new information needs arise over time, since they are only able to function over sets of documents prepared at design time [Brusilovsky2007h]. The *labor-intensive*, and thus *high cost* (with respect to labor hours), requirements imposed upon the creation of such content models, necessarily results in a limited volume and diversity of resources, available for delivery to AHSs [Conlan2002].

[4]This definition is inspired directly from that given by [Brusilovsky2007h], and is formalised in the glossary

In addition to the strong labor requirements, another reason explaining the low volume of resources available to individual AHSs, resides in the so-called *closed garden* nature of these content models. Resources produced for these systems lack interoperability between AHSs. They are only available within *proprietary formats* and are built for specific AHSs with *predefined reuse purposes* intended by the designers of each system (the AHS and content model designers are usually the same group of people) [Henze2002a].

Because resources available in closed corpus content models are built manually, a-priori of system deployment, this also entails the need to predict the type and quantity of resources, which will be needed by AHSs, based upon individual user needs [Steichen2011a]. This thesis defines an a-priori content supply approach as a supply model in which the content requirements needs of AHSs are *known* to the content supply model a-priori of any request, and resources are produced to match these requirements *at design time.*

There is an inherent dilemma regarding a traditional content supply production model for AHSs. On the one hand, AHSs aim to delivery experiences personalised to individual users, based upon various combinations of criteria, which themselves become increasingly unpredictable as the magnitude of possible user preference combinations increases. As was presented in section 2.2.3, in order to deal with this unpredictability inherent to AHS user needs, fragment variation techniques are already being used to reduce the need to know in advance what combination of needs individual user will require, and instead combine different elements of a page as these needs appear. However, this content is produced a-priori of system deployment, in low volumes and at a high cost. Hence, in order to maximise resource consumption (or return on investment), it also must be produced as ones-size-fits-all as possible, without affecting too much the personalisation potential deliverable by the AHS, so that it is reused as often as possible for many users. This involves a high risk in predicting in advance what will these common user needs be once the AHS is deployed. A risk which increases as the personalisation serves a wider range of users with more personal needs.

Hence, if the full potential of AHSs is to be brought to a mainstream audience, each of the issues concerning traditional content supply models, presented above, must be addressed.

2.3.4 Open Corpus Content Supply Models

In parallel to the development of traditional AHSs, a wealth of diverse information has now become accessible on the WWW as open corpus resources [Weal2007]. These resources can be viewed, from the point of view of educational AHSs for example, as large collections of learning material [Sosnovsky2012]. For many subjects, one can find online tutorials, examples, problems, lectures slides, etc. Hence, the reuse and incorporation of open corpus content could expand the range of adaptivity offered by AHSs. An open corpus of documents, consists of resources where documents, their attributes and relationships between documents and external models are unknown to AHS designers at system design time[5].

As presented in section 2.3.3 however, traditional AHSs can only operate upon a predefined set of limited resources, that have been manually structured and connected with proprietary domain models at design time. Open corpus resources available on the web on the other hand are very diverse. They come in different languages, cover an unrestricted set domains and are available in different formats. In order to leverage the full potential of resources available online, AHSs must hence manage to provide the same adaptivity delivered with closed corpus content, but instead with an unrestricted quantity of resources, of type and content unknown at design time. This section presents such systems, called Open Adaptive Hypermedia Systems (OAHSs), which attempt to deal with this problem.

2.3.4.1 Direct Open Corpus Reuse Content Delivery Models

As mentioned in section 2.3.3, the power of AHSs relying upon closed corpus content supply models resides in the fact that resources provided by such models are canned and known in advance at system design time [Knutov2009]. This allows AHSs to

[5]This definition is inspired directly from that given by [Brusilovsky2007h], and is formalised in the glossary

make full usage of known adaptation techniques (section 2.2.3) in order to provide the best personalised experience possible to individual users. Nevertheless, the level of manual labor involved in the creation of these resources along with the adaptation-specific information necessary to support these adaptation techniques, involves a great deal of effort, which limits the volume of such resources which can be created.

AHSs based upon an open corpus content supply model on the other hand, do not rely upon the creation of resources as such (since by definition they reuse existing content available on the web), which significantly reduces the amount of manual labor involved. Nevertheless, as for closed corpus content supply models, the development of such a content supply model requires the addition of adaptation-specific information to these web resources. Open corpus resources typically do not provide any element supporting the adaptive navigation and reuse of such resources [Steichen2011a]. In contrast with the closed corpus approach however, the resources to which this information must be added are unknown at design time. Hence the addition of such information can only be performed at run-time [Knutov2009].

For this reason, research carried out towards the development of OAHSs has so far focused as a priority upon the elaboration of techniques which support the addition of intra-hyperspace and indexing information to open corpus documents after AHSs have been deployed. Techniques currently used to achieve this purpose can be divided into three categories consisting of: i) user-supported, ii) keyword based and iii) concept based open corpus incorporation techniques. The selection of one technique for the purpose of augmenting open corpus documents with intra-hyperspace information does not necessarily preclude other techniques being used for the indexing of these documents and vice-versa. Each of these techniques support the incorporation of open corpus resources by separating the hypermedia content used as part of OAHSs from domain models and reasoning engines [Brusilovsky2007h], thus extending the trend towards component based architectures with respect to the underlying content model of AHSs (section 2.3.3).

User-supported incorporation techniques [Carmona2002, Henze2001a] have

been used by early OAHS to reuse open corpus resources within AHSs. The creation of a hyperspace network based upon a collection of independent open documents for example, as well as the indexing of these resources [Smith2003, Steichen2010b], can be achieved directly by *individual AHS users*. These techniques rely upon pre-existing AHS (based upon closed corpus content) as an underlying platform. This AHS is then augmented with appropriate linking and annotation tools needed by individual users to incorporate new relevant open corpus material to the AHS.

Two of the earliest systems using this approach consist of the SIGUE [Carmona2002] and KBS Hyperbook [Henze2001a] systems, in which both the hyperspace construction and indexing of open corpus resources is achieved by users of the AHS. In order to include additional open resources to the existing set, the AHS provide the ability for users to assign concepts relevant to each document incorporated as well as their type (introduction, problem statement ...). Links between resources are generated automatically (section 2.2.3) based upon the set of pages assigned to a concept, which happens to be related to the one browsed by an individual users. The sequencing adaptation (section 2.2.3) is thereafter carried out, independently of the resources used by the AHS (whether they are open or closed), and instead based upon the concepts contained within the domain model. An implementation of the KBS hyperbook within the Java programming language domain was carried out by Henze et.al, in which he shows how the Sun Java tutorial open corpus web pages could be easily incorporated with the proprietary resources used by the AHS (figure 2.6).

In a similar way as described in section 2.2.3, *community-based* techniques [Brusilovsky2004, Dieberger2002] have been also used to index open corpus pages based upon the activity (page annotations or traffic activity) of a predefined AHS user-community with respect to open corpus content incorporated within AHSs. As mentioned in section 2.3.1, since such indexing information is specific to the user activity occurring within a specific AHS and independent of the content being incorporated itself, the details of such techniques are not further discussed in this section.

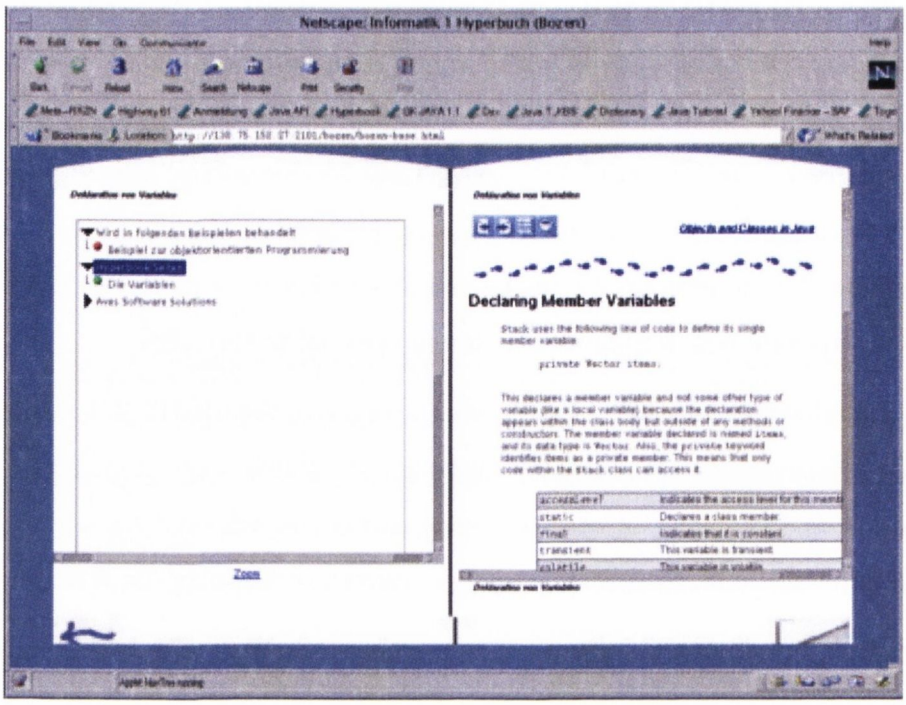


Figure 2.6: KBS Hyperbook [Henze2001a]

The reliance upon human judgement to incorporate open corpus resources within AHSs, means the selection, intra-hyperspace linking and indexing of such resources can be achieved with very high quality over a very large range of documents. This in turn would suggest resource sequencing techniques upon open corpus content can be performed with a precision equivalent to close corpus content. Nevertheless, the usage of such approaches can only be achieved as part of environments in which human-supported hypertext construction is possible (such as organisations or wiki-based environments), which limits the range of AHSs that can avail of this technique [Brusilovsky2007h]. Moreover, since this technique relies upon users of a predetermined platform, the AHS interoperability of adaptation-specific information produced for these resources is also questionable. Finally, and similarly as for a closed corpus content (section 2.3.3), although a content supply model based upon this technique does not rely upon the manual creation of resources, it nevertheless still requires a large amount of manual labor to be undertaken in order to link and index each open corpus resource. This drawback clearly limits the scale and volume at which open corpus resource can be reused within OAHSs. Ideally, AHSs should

be able to produce adaptation-specific information from open corpus documents without the help of a human indexer [Brusilovsky2001].

Hence, in order to achieve this goal, more scalable techniques that can automatically identify arbitrary attributes of open corpus resources have been developed. These techniques are very similar in essence to the content based sequencing adaptation techniques presented in section 2.2.3. ***Keyword incorporation techniques*** [Ahn2007, Billsus2000, Lieberman1995] for example represent such an approach. These techniques rely upon classical IR based keyword similarity algorithms [Zhou2008, Zhou2007a] in order to create links between similar pages. Within these systems, similarity metrics are calculated between each pair of open corpus resources to be incorporated. Resource similarity results achieving a score higher than an arbitrary threshold are thereafter connected by bi-directional hyperlinks. When coupled with keyword-level user-models, this incorporation approach can also be used to index open corpus resources with individual keywords, so as to support resource sequencing techniques (section 2.2.3). The ML-Tutor system [Smith2003] for example indexes open corpus resources using this approach. Each page incorporated within this platform is assigned a binary vector representing the occurrence or not of each keyword. These vectors are subsequently matched to a manually constructed keyword-based domain model, which assigns topics to individual pages. The system creates individual user models by combining keyword vector representations of pages browsed by users, which are thereafter used, along with the domain model, to recommend pages of topic not yet visited by each user.

In contrast with the previous approach, this open corpus incorporation technique is straightforward and fully automated. It can also be used to incorporate and index any textual open corpus resource into OAHSs so long as sets of keywords are previously assigned to domain and user model nodes. Although it has previously been successfully used in order to sequence resources based upon individual user interests [Ahn2007], its lower precision and inability to produce typed links or other adaptation-specific information of relevance to users (such as knowledge and goals) limits its usage within educational AHSs for example [Lops2011]. Furthermore, due

to the lack of typed links being produced, the resulting hyperspace structure created by this approach can become quite chaotic [Brusilovsky2007h].

Metadata based incorporation techniques attempt to go beyond the low-level keyword representation of open corpus resources and instead provide a semantically more accurate description of such resources. The key principle behind this approach is very similar to the keyword based equivalent and can be used both to link independent open resources together as well as for indexing purposes. Over recent years, many algorithms have been explored to automatically extract metadata from web resources (such as difficulty, narrative cohesion, interactivity type or interactivity level) [Sah2012, Sah2010, Hargood2011]. Such techniques are used in order to assign meta-data values to each open corpus resource incorporated within AHSs. Metadata similarity scores are calculated between each open resource and links are generated (section 2.2.3) for pairs obtaining a score above an arbitrary threshold. Concept based sequencing techniques (section 2.2.3) can thereafter be used to support the selection and presentation of individual resources based upon the metadata labels assigned. In contrast with keyword incorporation approaches however, this technique supports the structuring of the underlying hyperspace depending upon the level of complexity of chosen metadata schemes. The representation of open resources metadata as a part of an ontology for example [Sah2013, Sosnovsky2012, Steichen2011a, Ye2010, Yee2003] (figure 2.7), enables ontological reasoning to be used in order to compute similarity measures based upon the position of individual metadata within the ontology tree (as opposed to single metadata labels). The use of ontologies as part of the indexing approach also enables open corpus documents to be organised into hierarchies, which further improves the concept based sequencing navigation along the structure of these ontologies. [Ye2010] for example, incorporates open corpus content from the CiteSeer[6] website and automatically classifies individual resources based upon a predefined ontology. Additional ontological sub-categories are thereafter automatically generated from the set of open corpus resources in order to balance the distribution of documents per concept and improve the user navigation within this corpus.

[6]http://citeseerx.ist.psu.edu/

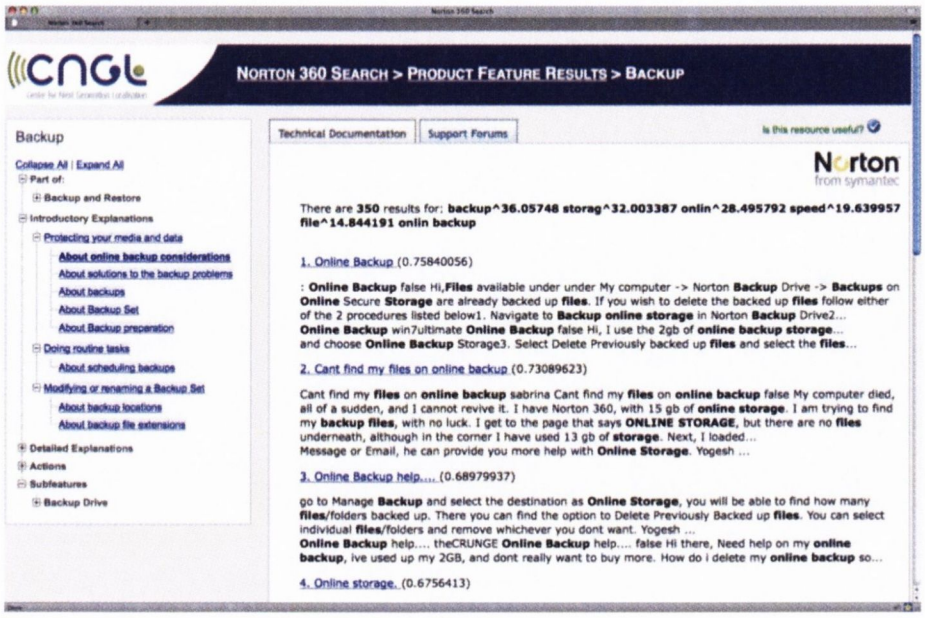


Figure 2.7: ARCHING ontology-based navigation [Steichen2011a]

As for the previous approach, once the metadata scheme is selected, this technique is fully automated. This means a large volume of resources can be processed for a given AHS. Also, since metadata expresses semantic similarity (as opposed to word level similarity), this approach provides a better link quality and structuring of the resulting hyperspace compared to keyword-based techniques [Brusilovsky2007h, Brusilovsky2007b] However, this technique sets strong requirements upon the metadata model selected for a given AHS [Brusilovsky2007b], which must be domain specific and possess established links (preferably of several types) between each concept. Also, if this open corpus incorporation approach is to be used as a basis for indexing, the same external model must be used for the user model in order to provide resource sequencing adaptation (section 2.2.3). Additionally, since the metadata generated from the open corpus resources is tightly associated with the metadata model/ontology chosen by an AHS, the interoperability of adaptation-specific information produced by these techniques is strongly dependent upon whether the given metadata model/ontology can be shared among AHSs and whether it is adequate for the domain environment selected by each AHS. Finally, depending upon the metadata generation technique selected, this approach requires

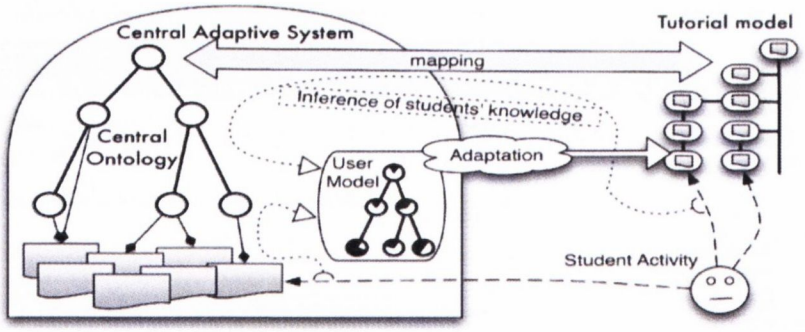


Figure 2.8: Open-corpus ontology mapping as part of the Ontology-based Open-corpus Personalization Service (OOPS) [Sosnovsky2012]

a decision to be made with respect to balancing the quality and range of metadata produced along with the degree of open corpus repository specificity upon which this resource incorporation approach will depend on. As described by [Sosnovsky2012], an extraction approach over-fitted to a single collection of open corpus documents will produce metadata usable only for this collection. In order to overcome this issue Sosnovsky et.al for example, propose (as part of the Ontology-based Open-corpus Personalization Service (OOPS) OAHS) to map ontology-based metadata models, extracted from each specific open corpus collection, to a predefined domain ontology (figure 2.8). Although the authors do not provide details on the mapping mechanism used within their OAHS, if confirmed in future research, this approach could reduce open corpus collection dependency for OAHS using this incorporation strategy.

2.3.4.2 Summary

OAHSs presented in this section all go beyond the closed garden resource limitations of traditional AHSs by pushing further the trend towards AHS component based architecture frameworks [Brusilovsky2001] (section 2.2.2),with a content model delegating all or some of the production and supply of content to external open corpus documents available on the WWW. In comparison with traditional content models, this approach provides a much larger range of resources to AHSs, available in a standardised web format (section 2.4.2.3) common to every AHS designer.

However, content supply models based upon the reuse of open corpus resources

(section 2.3.4.1) have so far focused in priority upon challenges related to the incorporation of these resources from the perspective of hyperspace construction and missing indexing information required to provide sequencing adaptation and overall hypertext navigation possibilities among independent resources. Although this represents, without any doubt, a fundamental aspect of the open corpus reuse challenge, the ability to support other adaptation techniques is also important.

Moreover, open corpus incorporation techniques presented within this section only supply such content in its native form, as one-size-fits-all document level granularity resources. For example, within the KBS hyperbook presented previously, once indexed and hyperlinked, open corpus resources are incorporated directly within the AHS, as in a traditional web browser, using an iframe (figure 2.6). Resource adaptation techniques were developed as ways to manage canned content fragments [Bunt2007], produced as self-contained resources with specific granularities (section 2.2.3). This document level granularity delivery restriction clearly reduces the extent to which resource modification or recomposition techniques (section 2.2.3) can be supported by OAHS. As pointed out by [Lawless2009], *there is an inverse relationship between the potential reusability of content and its granularity.* The reuse potential of a previously published news page, complete with original menus, advertisements and user comments, is far less adequate than if the article could be reused alone, de-contextualised from its original setting, at various levels of granularity (from a paragraph on a specific topic to the entire article). Although web open corpus resources are available in a common standard web format, resource modification and recomposition techniques would be better supported if open corpus resources were delivered to individual AHSs in a format of choice associated and with the metadata required by each system to support this adaptation. As pointed out by [Bunt2007], presenting open corpus resource in their native form does *"help to maintain the context of information displayed to users, however it can also generate information overload and reduce the attention towards the most relevant information, defeating one of the very reasons for having AHSs in the first place".* Some incorporation techniques are already augmenting open corpus resources with additional metadata usable by AHSs, however this metadata is typically tightly coupled with a predetermined AHS content

or domain model, which could limit adaptation capabilities of AHSs with different models, as well as interoperability between platforms [Brusilovsky2007h]. The large scale usage of open corpus within these systems, with appropriate metadata shareable between AHSs, would imply the necessity for some sort of open domain/content model common to the AHS community. Recent development in AHSs have already started modifying narrative components of these systems in order to incorporate open and large scale semantic sources such as DBpedia to enhance smaller proprietary domain ontologies [Steichen2011a].

Finally, it could also be argued that the process of indexing open corpus resources within AHSs could benefit from the ability to access such resources at different levels of granularities. Indexing portions of open corpus resources, as opposed to document level content, could support the indexing of conceptually self-contained fragments, thus easing further any recomposition adaptation. Hence, the ability for open corpus models to supply resources to OAHSs as right-fitted content packages could clearly improve its reuse within such systems.

2.3.4.3 Right-fitted Open Corpus Reuse Models

As presented in section 2.2.3, a considerable range of techniques used by AHSs to provide personalisation experiences to users rely upon the modification of resources and/or their recomposition. While performing such adaptation upon closed sets of resources is relatively straightforward (since each resource is known at system design time), performing this type of adaptation upon open corpus resources (of type unknown in advance) is much more challenging. Section 2.3.4.1 showed how open corpus resource supply models only deliver such content as document level resources, which limits the extent to which they can be modified or recomposed together. Hence, in order to provide the full range of adaptation techniques upon open corpus resources, available to the adaptive hypermedia community (section 2.2.3), this content must be decomposed into more granular, self-contained fragments, meeting specific AHS content requirement needs, with metadata supporting its modification and reuse for adaptivity purposes.

The PMCC AHS proposed by [Steichen2012b] for example, aims at delivering a personalised experience to individual users within the context of a customer care use case scenario. It does so using a metadata-based incorporation approach (section 2.3.4.1) of user generated open corpus resources (harvested from online web forums); adapted and recomposed together with corporate closed corpus product manual resources. As can be seen in figure 2.9, in addition to open and closed fragment recomposition, collapsible adaptation techniques are also used over open corpus resources.

This system provides a good demonstration of what is possible when open corpus resources are delivered in a form which supports recomposition and resource modification techniques. Open corpus resources incorporated in this platform however are right-fitted and fragmented using a wrapper-based algorithm. As explained in section 2.4.3.3, this content fragmentation approach produces individual fragments and metadata of high precision and quality. However it also requires the repositories, of resources to be processed, to be known in advance of system deployment and requires a substantial amount of manual labor (to adapt algorithms)

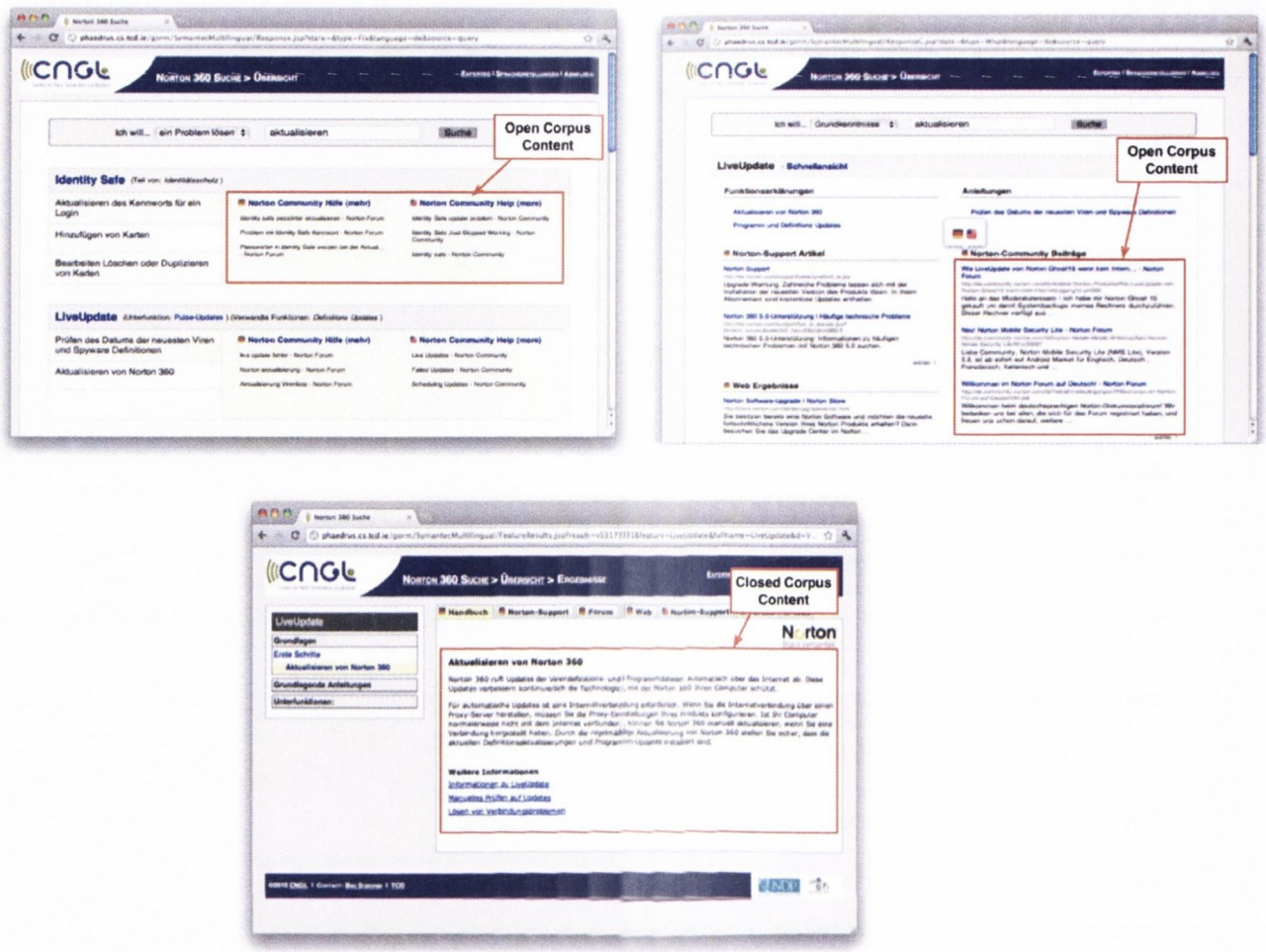


Figure 2.9: PMCC Open and Close content recomposition [Steichen2012b]

each time a new set of repositories are targeted for incorporation. The process used to generate additional metadata valuable for adaptation purposes relies upon the analysis and identification of pre-determined metadata clues contained within documents targeted. These limitations reduce significantly the range and volume of open corpus resources which can be incorporated and right-fitted for the purpose of AHS reuse (in comparison which techniques presented in section 2.3.4), which ultimately defeats the purpose of AHS open corpus reuse in the first place.

These techniques, in essence, attempt to deal with the challenge of open web corpus heterogeneity by reducing the degree of unpredictability and variability inherent to such resources, which limits the scale of reuse. As pointed out in section 2.3.4.1, the

fundamental difficulty in incorporating open corpus resources in AHSs resides in the fact that the type and quantity of these resources is unknown in advance, which entails the incorporation procedure must be achieved at run-time. Hence, the ability to right-fit open corpus resources from unknown repositories at run time should become an integral part of OAHSs, if the full range of adaptation techniques is to be applied upon this category of resources [Knutov2009].

Despite this limitation however, very little research has so far focused upon the automated adaptation of open corpus resources for reuse within OAHSs [Knutov2009]. This is not surprising, since the ability to adapt resources at run-time, outside the control of AHS designers, is quite challenging. In relation to the process of automatically structuring information, extracted from a set of unknown and unstructured open corpus resources, two related research areas consisting of text summarisation (specifically text summarisation extraction techniques) [Lloret2012, Lee2005] and information extraction [Schedl2008, Holzinger2006] come to consideration. With respect to the former, techniques developed within this field focus upon the extraction of pertinent keywords or sentences from open corpus documents with the ultimate aim to produce a more condensed synthetic piece of content depicting the essential conceptual properties of the original document(s) processed [Lloret2012]. The latter on the other hand, focuses in particular upon the extraction of data, as opposed to fragments, with the aim of providing a database type structure to unstructured information buried within open corpus resources. As a result, the product of both fields is better suited for integration within areas such as IR, question answering, and text classification. And as a consequence, the automated decomposition of open corpus resources into self-contained and AHS reusable fragments, is mostly an area which still requires to be investigated.

To the best of the authors knowledge, and at the time of writing, the closest example of a system performing run-time and right-fitted AHS incorporation of open corpus resources, consists of the ArtEquAKT system developed by [Weal2007]. By combining information extraction, knowledge management and adaptive document generation, the ArtEquAKT system creates dynamic biographies of artists by harvesting and

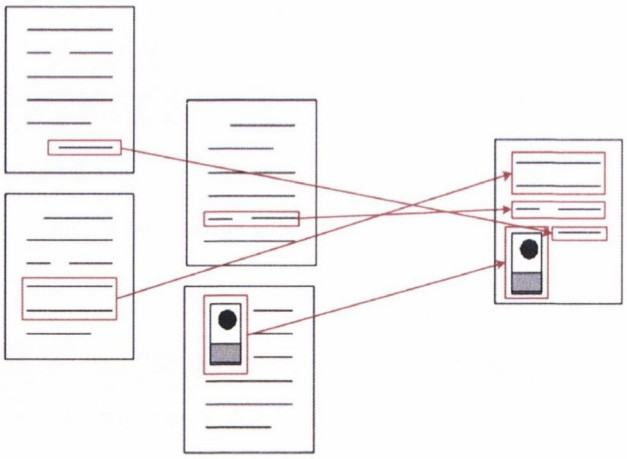


Figure 2.10: ArtEquAKT [Weal2007]

repurposing existing material on the web (figure 2.10).

It relies on traditional search engine to identify, harvest open corpus resources with biographical content, which are then indexed using a keyword-based incorporation technique (section 2.3.4.1). Each selected document is then divided into paragraphs. Sentences in each paragraphs are analysed syntactically to identify whether it contains any relevant knowledge about the artist requested by individual users. It relies on a pre-existing domain ontology produced specifically for the purpose of identifying relevant fragments of pages to reuse. Biography pages are then generated by rearranging relevant text fragments within biography templates, using a fragment variant approach (section 2.2.3).

In contrast with the previous AHS, the incorporation system used by the ArtEquAKT does not constrain open corpus pages to originate from a predefined repository known at design time and therefore provides a much wider range of open corpus resources to reuse. It enables the right-fitting of open corpus resources at run time, without any knowledge of the type of resources it is about to target, which matches the overall goal of OAHSs. However, the reuse potential of open corpus resources processed by this system is tightly coupled with a very narrow predefined use case scenario (i.e artist biographies) and recomposition template style (lists of relevant paragraphs). It provides little or no metadata which could be of use to other AHSs in order to adapt or recompose generated fragments in ways and for purposes of their own. Fragments

generated would be of little use and would not support the kind of content adaptation performed by the PMCC system presented earlier for example. Finally, in contrast with the high quality fragments generated by the PMCC approach, although the reuse of open corpus resources in the form of single paragraphs in the selected use case scenario might be adequate, fragments generated by this technique are produced without any consideration with respect to the original context or structural and semantic cohesion of the document it originated from, which is problematic if such fragments were to be reused by other AHSs (see section 2.2.3).

2.3.4.4 Summary

The reuse of open corpus resources by the AHSs presented in this section, clearly demonstrate the need of these systems to go beyond the document level open corpus reuse paradigm, presented in the previous section, and to instead reuse these resources at different levels of granularity and with adaptation specific information. They also demonstrate how the process of right fitting these resources can support the reuse of these resources within more advanced adaptation scenarios.

However, these approaches are, what this research defines as, content and/or AHS specific techniques. Within this thesis, content/AHS specific techniques refer to reuse approaches in which i) specific open corpus repositories, known at design time, are targeted for reuse in order to serve ii) predetermined niche content requirements for iii) an established set of AHSs.

Although these right fitting techniques do improve the reuse of open corpus resources within AHSs, in comparison to direct open corpus reuse techniques (see previous section), they do so by producing resources that serve the reuse needs of predetermined AHSs (usually one). Hence, in a similar way as for closed corpus content models, right-fitted resources lack interoperability between AHSs. Moreover, as they require open corpus repository targets to be known in advance, the range of open corpus resources reusable through these techniques is clearly reduced in comparison to direct reuse approaches. Additionally, whenever a change in resource repository targets or reuse purposes occurs, algorithms developed to support these techniques must be altered accordingly or replaced altogether, which can be quite labor intensive.

Hence, although these techniques address limitations inherent to the direct reuse of open corpus resources within AHSs, they do so by increasing the reuse potential of only a selected group of such resources, while simultaneously reducing the overall of all open corpus resources available for reuse by the AHS community.

2.3.5 Open Corpus Identification and Harvesting Techniques

Although the incorporation of open corpus resources within AHSs (to support adaptive techniques) is vital for the future development of OAHSs, the process which involves the identification and harvesting of these resources in the first place is also critical. Properly incorporating open corpus resources is pointless if the resources themselves are of no relevance to the reuse scenario selected by a given AHS.

As the identification of web resources is a vast field of its own extensively studied, this section aims to present the reader with an overall view of the range and diversity of techniques available to the community. In particular, this section focuses upon the techniques used so far by OAHSs as well as those that could provide benefits to the adaptive community in the future. For a deeper insight into each of the individual fields mentioned in this section, the reader is referred to reviews carried out by [Ghorab2012], [Micarelli2007b] and [Steichen2012c].

2.3.5.1 Standard Identification and Harvesting

To the best of the author's knowledge, research investigating OAHSs has so far mostly focused upon the problem of correctly incorporating open corpus resources into existing AHSs. The process of identifying and harvesting these resources from the web for AHS reuse, has therefore not received as much attention in the literature as the former. Nevertheless, whenever this topic is mentioned, the vast majority of OAHSs developed until recently, have mostly relied upon either the manual identification and harvesting of resources, or the use of standard IR mechanisms to do so.

When manual identification of resources is carried out, this is usually performed by individual users of the AHS and is closely intertwined with the resource incorporation process presented in section 2.3.4. The SIGUE [Carmona2002] or KBS Hyperbook [Henze2002a] OAHSs for example, provide the ability for tutors to incorporate any web resources of their choice within an existing AHS. However they do not provide

any support for the identification of relevant open corpus resources and solely rely upon users to individually identify and select appropriate content to be reused. While this approach is straightforward, it clearly suffers from similar drawbacks encountered with user-based open corpus incorporation techniques (section 2.3.4.1). Due to the amount of manual labor involved, the reliance upon individual users limits the scale at which relevant open corpus pages can be identified and harvested from the web in terms of volume and speed.

For this reason, automated standard IR and web crawling techniques have been also used to perform this task. ArtEquAKT [Kim2002b] for example relies upon IR based identification and harvesting of open corpus resources using standard search engine systems. The Personal Reader [Dolog2004] on the other hand uses Lixto [Hassan2005] for standard web crawling and Edutella [Nejdl2002] a peer to peer search system, to perform the identification and harvesting of resources. The use of fully-automated techniques to achieve this task enables much larger volumes of resources to be identified for the purpose of reuse within OAHSs. Moreover, standard IR or web crawling techniques are relatively easy to integrate with AHSs, since they simply rely upon a query to be submitted as an input and return an ordered list of pages ranked according to a particular matching algorithm. However, since traditional IR systems only rely upon a set of keywords as an input, they fail to take into account any additional information available within AHSs such as user, knowledge or domain models used by AHSs and can therefore only serve immediate needs expressed by users. Hence, two users, with similar interests but different knowledge backgrounds or preferences for example, would obtain the same results following the submission of the same query [Micarelli2007b]. The models available within AHSs could clearly be of high value within the context of automated identification and harvesting of relevant open corpus resources for these systems.

2.3.5.2 Personalised Identification and Harvesting

Several approaches, which aim to modify the selection of resources returned by open corpus search tools, based upon individual users or groups of people, have been elaborated and used within more recent OAHSs. In order to deliver relevant

resources, traditional IR systems typically rely upon two phases to pursue this aim. The first phase requires them to crawl the web by following links available within pages, gather any resources encountered and assign to each one of them a document model representation. The second phase thereafter matches individual queries submitted by users with the most relevant resource previously discovered by the system. Techniques providing tailored result sets, can hence be divided into two categories based upon whether they affect the first or second phase of this open corpus identification process.

Topical crawling techniques, also referred to as ***focused crawling***, for example, attempt to tailor the crawling phase of IR systems. They are based upon the notion of topical locality which refers to the observation that web pages are typically linked to other pages with semantically similar content. In other words, web pages tend to link to other pages which contain related information. Focused crawlers exploit this phenomenon to aid the topic-oriented discovery of web pages [Lawless2009]. Unlike a standard crawler, which traverses the web downloading all the documents it comes across, a focused crawler is developed to retrieve only documents related to a given topic of interest [Micarelli2007b]. Focused web crawlers enable OAHSs to refine the set of open corpus resources considered for incorporation. The range of focused crawling techniques investigated by the research community is very large. They can be divided into adaptive or non-adaptive categories based upon whether they can take into account any feedback retrieved from pages crawled or the users of AHS initiating the crawl. Non-adaptive focused crawlers in essence function as classifiers with a learning phase ending before a search process is started. The most popular techniques used by this category of topical crawlers include link prioritisation [DeBra1994a], taxonomic [Chakrabarti2002, Chakrabarti1999], context graph [Diligenti2000] and tunnelling crawling [Bergmark2002] techniques. Adaptive alternatives, such as intelligent crawling [Aggarwal2001], genetic algorithm [Menczer2000] and ant-based crawling [Gasparetti2004] on the other hand, adjust the learning instead during the crawling itself, according to a particular environment and/or relationship with input parameters. While non-adaptive focused crawlers are suitable for OAHSs serving predictable broad content requirements, adaptive versions on the other hand offer

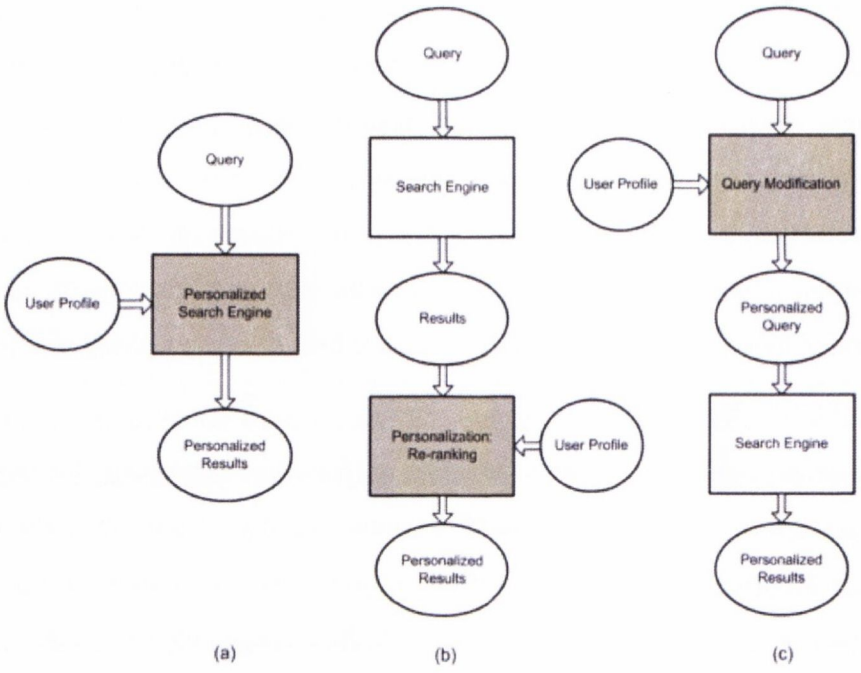


Figure 2.11: Information retrieval personalisation approaches [Micarelli2007a]

OAHSs the ability to provide requirements at run-time instead [Micarelli2007b]. This enables such AHSs to learn and be responsive to potential alterations of user needs representations, while the platform is deployed, and hence direct the crawling of additional resources accordingly. The educational OAHS presented by [Steichen2009a] for example, built on top of the APeLS AHS [Conlan2004a], uses a topical crawling approach to gather educational material available on the WWW. In order to achieve this task, it uses the autonomous Open Corpus Content Service (OCCS) focused crawler, which enables the discovery, classification, harvesting and indexing of content from open corpus sources [Lawless2008a]. This crawler can be used to create domain specific caches. The use of this automated service enables the AHS to ensure that large volumes of open corpus content, highly relevant to the learning domain targeted, will be available by the platform for incorporation.

Once a set of open corpus resources has been crawled, the process which involves matching user needs with individual resources, discovered by the crawler, can also be personalised. Such methods, referred to as ***personalised information retrieval*** techniques, aim to provide search results that are not only of relevance to a query but

are also of particular relevance to the user who submitted the query [Ghorab2012]. These techniques attempt to overcome issues emanating from the low level of interactions involved by individual users with searches [Jansen2000], as well as queries being in general too short (with an average of two to three keywords per query) [Beitzel2004], incomplete or ambiguous due to polysemy [Micarelli2007a]. An abundance of methods, using both explicit and implicit feedback from users, have been developed for each of these individual categories. User information taken into account by Personalised Information Retrievals (PIRs) can consist of user browsing history [Sugiyama2004], user attributes (age, gender, education etc) as well as usage data (activity of the user during the search) [Kelly2003].

These techniques can be divided into three subcategories based upon the point at which the personalisation occurs during the search. Established one-size-fits-all ranking algorithms, for example, such as the HITS [Kleinberg1999], PageRank [Brin1998] or BM25 [Robertson1994], can be modified in order to take into account and incorporate various aspects of individual users [Tanudjaja2002, Teevan2005] (figure 2.11a). [Haveliwala2003], for instance modifies the PageRank algorithm by precomputing a set of PageRank vectors for each Open Directory Project (ODP) category taken into account. Individual IR queries are then compared with each of the topical ODP categories selected, in order to produce a linear combination of PageRank vectors. The personalised ranking thereafter consists of the combination of each PageRank calculation, weighted according this linear combination. In contrast with the former, a second category of PIR techniques [Speretta2005a, Stamou2008], which do not modify existing one-size-fits-all rankings produced by search engines has also been investigated. These approaches instead rerank the output of search engines based upon its combination with personalised ranking data collected for each user (figure 2.11b). [Stamou2008] for example, map past user queries with a topical ontology. Topics/query relations available within the ontology, are then used to rerank standard search results into a personalised reranked version. Finally, while the two previous approaches attempt to incorporate user profile information during the document retrieval process, a large array of techniques based upon query modification and expansion can also be used to alter the query submitted by a user

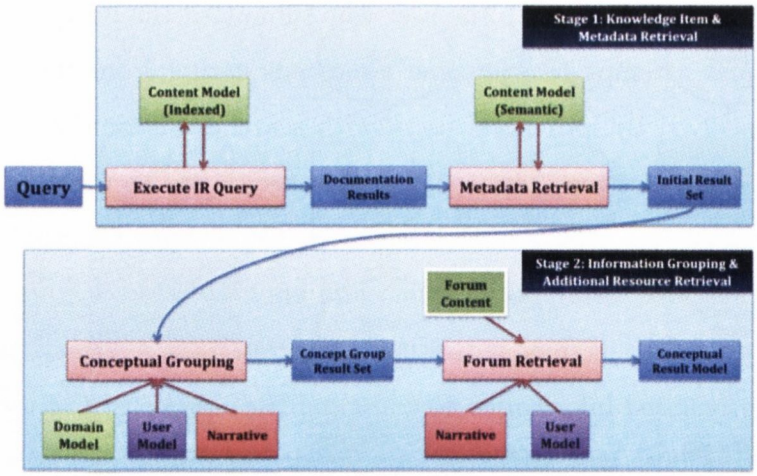


Figure 2.12: PMCC Personalised Information Retrieval [Steichen2011a]

before reaching search engines [Maxwell2013, Babashzadeh2013, Gao2013]. Multiple additional queries for example can be submitted on behalf of a user, or individual user queries can also be augmented (or expanded) based upon information collected for each user [Mangold2007]. [Gao2013] for example, represents search logs as labelled directed graphs. Expansion terms added to the original set of keywords are then selected based upon path-constrained random walk probability calculations for each user query submitted. PIR techniques have already been used in OAHSs to tailor the selection of open corpus resources. [Millard2003] for example use an automatic Wordnet[7] based query expansion approach in order to identify open corpus resources on the web and provide adapted biographical descriptions of selected artists to users. The PMCC AHS [Steichen2011a] on the other hand, uses a combination of query expansion and multi-query initiation techniques in order to select web forum resources to be incorporated within its platform. This hybrid AHS combines closed and open corpus resources into a single user experience, by augmenting an initial user query with data retrieved during a closed corpus search (figure 2.12). Concepts, assigned to closed corpus resources returned to a user, are extracted and grouped together. For each concept group retrieved, additional expanded queries are triggered using as a basis for expanded terms, the textual content of top ranked closed corpus resources.

[7]http://wordnet.princeton.edu/ [Accessed: October 3, 2014]

2.3.5.3 Summary

Since the birth of the WWW, the field of IR has generated an extensive array of techniques aimed at improving the identification of large volumes of web resources relevant to individual users. Personalised alternatives to traditional IR techniques can also take into account individual characteristics of users, to further refine the relevance of resources identified. The benefits offered by these techniques clearly matches the needs of OAHSs, which already possess detailed models of individual users and seek to obtain large volumes of relevant open corpus resources. Although a few pioneer OAHSs have already started using these techniques, if AHSs are to be deployed at a large scale to a mainstream audience, IR systems will clearly become a fundamental component of such platforms. Open content supply modules or services, supporting the needs of these systems, should hence be designed in order to take into account the diversity of existing and upcoming techniques offered by this field.

However, despite these benefits, such systems only delivery open corpus resources in their original form, structured for its original purpose, which mostly is aimed at human reading. In most cases this is more than adequate, however, as presented in section 2.3.4, for the purpose of AHSs reuse, resources to be incorporated must be available at different levels of granularity, with metadata supporting its adaptive reuse [Weal2007]. Resources are only delivered by these systems as one-size-fits-all document level packages, with very limited metadata, regardless of the receivers reuse intentions. Even personalised alternatives only focus upon the tailored harvesting or retrieval of these resources without modifying the original document [Steichen2011a].

2.3.6 Content Supply Discussion

This section presented various approaches used in order to produce, deliver and incorporate resources within AHS content models. Closed corpus, direct and right-fitted open corpus reuse techniques, each, in that order, attempt to go beyond its supply content model predecessor's limitations. Among all content model approaches presented, three cross-cutting characteristics, are recurring concerns among techniques presented. The degree to which resources provided to an AHS i) need to comply to any pre-determined reuse conditions when selected for reuse (either directly based on the content model requirements of an AHS or based upon the type of resources which can be processed and right fitted by a AHS), ii) are right fitted to particular AHS content requirements and whether iii) these content requirements must be determined in advance of system deployment or not, indirectly impact the volume of resources ultimately delivered by an approach.

With respect to the first condition, direct open corpus reuse content models offer the least restrictive approach since by definition they can incorporate any resource available on the WWW without any modification. This enables the wide range of resources available on the web to be reused as part of AHSs, which rely upon this type of content model. Closed corpus content models, on the other hand, represent the most restrictive approach since only content, which adheres to an arbitrary AHSs format, can be considered for incorporation. The reverse is true however with respect to the degree to which resources, supplied by these content models, are right fitted to specific content requirements of individual AHSs. While content used within closed corpus content models is produced specifically for a predetermined AHS (and hence can be created directly with the intention of matching content requirements), content provided by direct open corpus reuse content models, offers the lowest level of right-fitting since resources are delivered as they exist (as document level resources), created most of the time for purposes other than that sought by a particular AHS. Content models based upon the incorporation of open corpus resources right fitted to particular content requirements of AHSs on the other hand, provide a trade-off solution between the two previous content models. They improve right fitting aspects

of resources incorporated, in comparison with direct open corpus reuse, at the price of a lower targetable resource range.

However, none of these content models can perform the complete process of identifying AHS content requirements and incorporating novel resources on an ad-hoc basis. Because all of them, rely at some point upon some form of manual labor, niche content requirements of AHSs must be known in advance of system deployment. This implies content model designers can predict in advance what these requirements will be. The only exception would be a content model, which relies upon the direct reuse of open corpus resources based upon keyword identification and incorporation techniques. However as mentioned in section 2.3.4.1, the level of adaptation offered by this incorporation technique is limited. As the diversity of individual user needs increases, so does the difficulty in predicting niche content requirements of AHSs (section 1.1). The ability to identify these content requirements and supply adequate resources on an ad-hoc basis therefore also increases. Moreover, current information requests on the web are also more dynamic and ad-hoc in nature [Steichen2011a], which further pushes AHS content models to support this need.

The only approach, in which a large diversity of AHS content requirements could be served with a large volume of resources, would therefore involve right-fitting open corpus resources on demand, only after specific AHS content requirements are made explicit to the content model. The following section hence presents the reader with various state of the art content reuse and analysis techniques, which could serve as a basis for the improvement of AHS content models and supply systems.

2.4 Content Reuse and Analysis Techniques

2.4.1 Introduction

Section 2.2 and 2.3.4 presented an overview of AHSs in general, their anatomy, the adaptation techniques they rely upon along with the various content models used to support these systems and their limits.

This section on the other hand, explores more general aspects of content. The aim is to identify techniques and technologies, which could be of use with respect to the aim of improving existing content models described previously. In particular, section 2.4.2 presents various forms of reuse which have been used successfully in the past to maximise the value of existing resources within a community. The various reuse approaches described in this section will serve as design inspirations for the new content supply service designed in the following chapter. Section 2.4.3 and 2.4.4 on the other hand, present techniques used to analyse properties of open corpus resources. As mentioned in section 2.1, techniques used to analyse structural aspects of resources are explored in more detail since this content analysis aspect was identified at an early stage in this research as critical for the development of the content slicing approach proposed by this research.

2.4.2 Content Reuse Mechanisms

2.4.2.1 Introduction

The production and delivery of content has traditionally been a very linear process [Lawless2009], involving the painstaking authoring of content by a domain expert [Meyer2011], for predetermined purposes and consumer types. Section 2.3 described how this process has so far limited the widespread adoption of AHSs, leading to the reuse of external resources as an answer to these restrictions. This section hence, presents an insight into broad techniques, currently used to support and improve the reuse of resources in general. Each of these techniques improves particular aspects of reuse and are divided accordingly into three forms; namely reuse through i) encapsulation, ii) shared publishing/distribution mechanism and iii) repurposing. The aim is to provide an overall insight into each technique and also provide a basis for inspiration in the AHS content supply model improvement design and implementation decisions presented in chapter 3 and 4.

2.4.2.2 Reuse Through Encapsulation

Although the notion of "content reuse" does not necessarily preclude the event involving the reuse of a resource by its original author, in most cases however it is intrinsically linked with the idea of sharing, which assumes the actors involved in the production and reuse of a resource are different from each other. This is the case in particular with the form of reuse, which involves content encapsulation. The aim of this technique is to go beyond the traditional bespoke, proprietary content limitations of traditional production and delivery methods, by improving its *discoverability and interoperability* between potential content consumers [Lawless2009]. The diversity of ways in which a piece of content can be described or presented to users, can seriously reduce the number of consumers capable of reusing this resource down to only those who strictly adhere to both similar content descriptions and formatting requirements selected by individual authors. For this reason, the process of encapsulating content, involves the production of resources according to standard content models and formats.

The field of eLearning in particular provides a good example of content encapsulating techniques used to improve the reuse interoperability of learning resources produced across educational institutions. Popular content modelling standards such as Learning Object Model (LOM)[8], Dublin Core [9], IMS Learning Resource Metadata[10] or Sharable Content Object Reference Model (SCORM)[11] aim to support the creation, by content authors, of precise definitions of individual resources, along with information regarding how each can be reused and for what purpose. The LOM content model for example, is divided along nine separate categories, each describing different aspects of individual eLearning resources. General information (e.g. title, description or language used), technical (e.g: format used and size of resource) or copyright related categories for example describe overall aspects of these resources, while educational (e.g. modes of learning, age range or user interaction) or relation categories (prerequisite resources) cover more domain specific characteristics. SCORM on the other hand, represents resources based upon three sub-specifications. Content packaging specifications for example specify how resources should be packaged and described (e.g: through the use of Extensible Markup Language (XML) and LOM into so-called Sharable Content Objects (SCOs)), while run-time and sequencing specifications respectively describe how resources should be launched and how learners can move across SCOs. The use of encapsulating techniques to improve the reuse of content has already been used by AHSs such as the APeLS system [Conlan2004a], which uses LOM as a metadata standard for its content model. The encapsulation of resources also makes it possible to build large repositories of such resources, in which content producers and consumers are able to store and retrieve relevant content [Henze2002a]. Examples of such repositories include Multimedia Educational Resource for Learning and Online Teaching (MERLOT)[12] or National Digital Learning Resources (NDLR)[13].

Hence, content encapsulation standards define and structure the metadata and

[8]http://ltsc.ieee.org/wg12/ [Accessed: October 3, 2014]

[9]http://dublincore.org/ [Accessed: October 3, 2014]

[10]http://www.imsglobal.org/metadata/ [Accessed: October 3, 2014]

[11]http://scorm.com/scorm-explained/technical-scorm/ [Accessed: October 3, 2014]

[12]http://www.merlot.org [Accessed: October 3, 2014]

[13]http://www.ndlr.ie/ [Accessed: October 3, 2014]

formats associated with each resource. This allows a common structure and descriptive vocabulary to be used across resource producers and/or consumers, which supports the discovery, consumption and reuse of resources from various origins by each institution. The use of content encapsulation techniques however, also involves a dilemma between on the one hand, a high metadata accuracy required for each resource and on the other hand, the level of complexity and manual labor involved in its production [Bailey2006]. As the volume of resources produced by an institution grows, so does the amount of manual labor required to describe and structure these resources. Automated metadata generation techniques thereafter become necessary. However as described in section 2.3.4, the accuracy and range of metadata produced by these techniques also decreases.

2.4.2.3 Reuse Through Shared Publishing and Delivery Mechanisms

The previous section showed how the process of encapsulating resources, as well as making them available in large resource repositories, aimed at improving their interoperability across content consumers, and therefore going beyond the closed garden limitations inherent to proprietary content production approaches (section 2.3.3). These large repositories however can still be seen as "closed pools" of interoperable resources, since the resource publication and delivery mechanisms are specific to each repository. Reusability is not only a property of a resource, but also of the system, which supports its discoverability and reuse [Meyer2011].

Reuse approaches, using standard distributed publishing and delivery mechanisms, attempt to go beyond this limitation by providing a *common publishing and delivery mechanism* across resource repositories. The WWW and linked data environments represent, to date, the most successful examples of this type of reuse.

The success of the WWW, originally developed by [BernersLee1994a], relies upon its operability between different machines, ease of use and the fact that it is built on open standards. Documents can be instantly published to the WWW and, potentially, shared with millions of individuals worldwide at essentially no cost [Lawless2009]. It relies upon four core technologies [Sah2009a], namely i) a HyperText Markup

Language (HTML) markup language, which encapsulates resources with a common human readable format, consumable by ii) a web browser; and iii) a universal resource addressing system (Universal Resource Identifier (URI)) coupled with iv) a network protocol for web servers (HyperText Transfer Protocol (HTTP)), which respectively standardise the publishing and delivery of resources between content producers and consumers. URIs consist of unique short strings, representing the location of each resource on the web. This allows content producers to publish resources, identifiable by the community, through a common addressing scheme. HTTP [Fielding1999] on the other hand, is a protocol based on the internets Transmission Control Protocol (TCP) [Cerf1974], which allows the standardised transfer of documents between content publishers and consumers.

While the WWW aims to improve the reuse of resources in the form of documents, linked data [Bernerslee2006] (also sometimes referred to as the web of data and part of the larger semantic web vision [BernersLee2006c]) on the other hand, focuses instead upon improving the reuse of machine readable data. It follows however the same principles as the WWW and is based upon three core open standard technologies, namely: i) Resource Description Framework (RDF) to encapsulate data relationships, ii) URIs to identify individual data or relationship exposed as resources and iii) the use of HTTP to retrieve RDF data associated with each single URI RDF [W3c2004a] is a metadata model used to make statements about resources in the form of subject-predicate-object declarations called triples. URIs are used to represent each subject, predicate or object. This enables each statements to be shared between machines. RDF data is stored in repositories called triplestores (section 4.3.3.1) and can be queried using the SPARQL Protocol and RDF Query Language (SPARQL) protocol. As with HTML, RDF can link different datasets together (figure 2.13), which enables relationships between resources to be reused across repositories. Moreover, ontologies, which describe sets of concepts and relationships between them, can also be used as a common RDF vocabulary, between repositories for publishing data. Many ontologies such as Friend-Of-A-Friend[14] or Simple Knowledge

[14]http://www.foaf-project.org/ [Accessed: October 3, 2014]

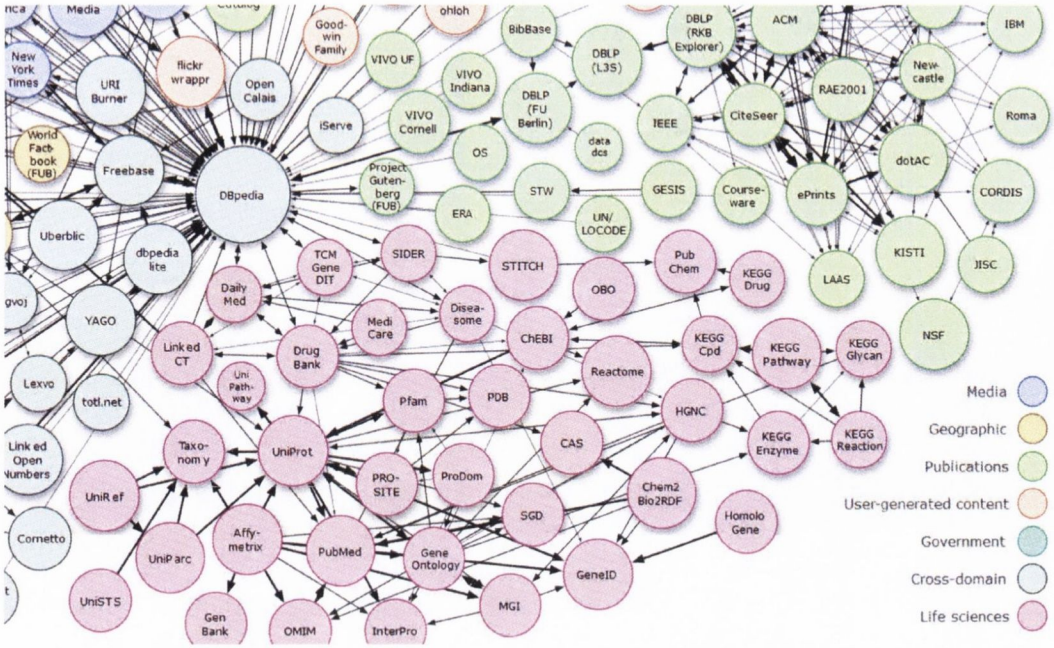


Figure 2.13: Portion of the LinkedData cloud as of 2011[16]

Organisation Systems[15] exist nowadays. They enable the automated inference of new relationships between resources.

In essence, this form of reuse improvement enables the collection of distributed closed resource pools (as described above), to be accessed by content consumers as if they consisted of a single unique pool of interoperable resources, regardless of the repository in which they are stored. It removes the need for content consumers to adjust to each content publication and distribution mechanism used by individual repositories. The value derived from this form of reuse however is directly related to the rate of adoption of selected standards (which in turn encourages openness). It requires the initial support of a community willing to promote and bootstrap this resource sharing mechanism up to a critical mass, which can represent an important deterrent. Many attempts to create the equivalent of the WWW such as Microcosm or Hyper-G [Sah2009a] for example, have previously failed to disseminate to a mainstream audience. For these reasons, the ability to support this type of reuse through open standards, and dedicated communities to increase the awareness and usage of these principles is fundamental.

[15]http://www.w3.org/2004/02/skos/ [Accessed: October 3, 2014]

2.4.2.4 Reuse Through Alteration

The reuse approaches described in the two previous sections involve either the use of standardised metadata or publishing/delivery protocols in order to respectively describe and transfer resources targeted for reuse. Non of these approaches however, involve the modification of the original resources themselves. Although the reuse of a resource, which does not involve its modification, can potentially include reusing it for purposes not originally planed by its author, the inability to modify it however, certainly limits the range of purposes for which it could be reused for. Nevertheless, the reuse of existing resources is mostly performed within ranges of reuse scenarios close to those predetermined by original authors (i.e. being read by students for example or inserted as it is as part of a larger corpus of resources). However, besides being discovered, accessed and delivered for reuse with other resources, existing content may also be seen as a preliminary product of an authoring process [Hodgins2002, Meyer2011]. Reuse approaches which involve modifying an existing resource hence aim to *increase the repurposing capabilities* of a resource, by adapting it to a new context not initially intended by its original author(s).

[Meyer2011] for example, follows this principle in the area of educational resources with the Repurposing Tool Suite prototype application proposed. This tool suite serves as a platform for different repurposing tools, such as modularisation, aggregation and adaptation tools, which users can utilise in order to improve the modularity of existing resources and in particular their multi-granularity. Existing resources are used as preliminary products for the creation of new learning resources, which accelerates the manual production of new learning objects for web based training. [GonzalezBarahona2006] also present the Edukalibre architecture, an open collaborative platform to support the free reuse and modification of educational resources in university settings. Edukalibre takes inspiration from lessons learned in the open software development communities and provides existing as well as new collaborative web based tools (such as the COLLAB and ConDOR web-based collaborative editing tools) to support the community-driven development of educational content. While the task consisting in the repurposing of existing resources

is performed in the previous example manually, automated alternatives have also been elaborated. The ArtEquAKT system developed by [Weal2007] (described in further details in section 2.3.4.3) for example, automatically repurposes parts of web resources as preliminary content in order to deliver a platform to users providing biographies of arbitrary artists. In this example, all of the content presented to users originates from various resources separately originally authored as stand alone documents and repurposed for the purpose of recomposition within an independent platform. Other approaches [Ahmadi2008] have also used this approach to automatically modify web pages originally authored for desktop presentation for the purpose of making them more readable on small handheld devices. Although the notion which consists of altering preliminary resources in order to produce new ones, is a powerful idea as it enables the ability to unlock the full potential of resources already produced, as is explained in further details in section 2.3.4.3, the difficulty involved in developing a system which can perform this task automatically at a large scale, over resources of various types and for various purposes is a challenge which is still being investigated.

2.4.2.5 Conclusion

This section presented the reader with three different forms of content reuse, which have proven successful in improving the reuse of existing resources with respect to various dimensions of reuse. If properly leveraged, these proven techniques could clearly benefit the adaptive community with respect to the existing content model and supply approaches used by AHSs. The following sections continue exploring various approaches and technologies available to the community, which could also be leveraged for same purpose. In particular, these sections focus more specifically upon tools available to conduct various types of analysis of open corpus resources.

2.4.3 Content Fragmentation Techniques

2.4.3.1 Introduction

As mentioned in section 1.1, prevailing approaches involving the reuse of open corpus content within AHSs suffer because they only provide limited control over the granularity of such resources. The ability to move beyond one-size-fits-all delivery represents therefore a critical aspect of open corpus reuse improvement. This section thus presents a review of content fragmentation algorithms, which aim to identify and extract coherent regions[17] of individual pages [Sleiman2012].

2.4.3.2 Overview of Content Fragmentation

Fragmentation techniques defer significantly from text segmentation algorithms, such as text-tiling [Hearst1997] or affinity propagation segmentation [Kazantseva2011]. While the latter techniques aim to segment texts (as opposed to complete web documents) into sub-topics, content fragmentation algorithms attempt instead to identify structurally coherent fragments of a document.

Content fragmentation techniques[18] are increasingly used as preliminary processing tasks within information extraction systems [Antonacopoulos2007a]. For this reason, approaches available in the literature are, most of the time, described not as stand-alone algorithms but instead as part of larger systems. While information extractors focus on extracting and structuring data records and their attributes, content fragmenters instead focus on identifying fragments of resources that contain this information [Leoncini2012, Sleiman2012].

Over time, page layouts have become increasingly complex. They have evolved from flat documents to today's deep nested structures, with sidebars, navigation panels, headers, advertisements etc. [ChakrabartiD2007a]. The reliance of HTML tags

[17]Formal definitions of what constitutes structural fragmentation and regions of a page are available within the glossary

[18]Content fragmentation is also referred to as structural fragmentation within this thesis. Other researches in the literature also refer to these approaches as region extractors, page segmenters or semantic structure analysers

within web documents, to detect relevant parts of pages, is increasingly erroneous [Mehta2005]. Modern web designers, for example, are moving towards the use of Cascading Style Sheets (CSS) to represent visual information, using mostly div tags to distinguish different sections of a page [Kohlschutter2008a]. Moreover, templates[19] represent a significant proportion of data on the web (between 40% and 50%) growing at a rate of approximately 6% per year [Gibson2005]. This non-informative content can affect web mining [ChakrabartiD2007a, Yi2003] or web search accuracy [Moura2010]. For this reason, such content must commonly be discarded as part of preliminary processing tasks.

Content fragmentation techniques are thus increasingly used for purposes such as web mining [Alassi2012, Leoncini2012, Hogue2005], web search [Cai2004, Moura2010, Mehta2005], document archiving [Esser2012] and classification [Hu2000] as well as content adaptation aimed at mobile browsing [Hattori2007, Yin2004, Yin2005, Ahmadi2008] or web accessibility [Mahmud2007].

This section serves as a basis, in section 3.3.3, for the selection of a structural fragmenter to be incorporated as part of the slicer implementation carried out for the purpose of this research. As numerous attempts to achieve this task have been undertaken in the past, this review aims not to provide an exhaustive list of all algorithms available in the state-of-the-art but instead to present an overview of general approaches used for achieving this purpose along with their implications and trade-offs. The review separates approaches, examined as part of this thesis, into five categories consisting of: i) content wrappers, ii) visual fragmenters, iii) Document Object Model (DOM)-based fragmenters, iv) template-based fragmenters and v) graph-based as well as semantic fragmenters. This taxonomy represents only one out of many possible categorisations of techniques available within this field [Laender2002a, Sleiman2012]. Also, as many algorithms borrow characteristics from various approaches, these categories are only intended to provide general trends to the reader as opposed to exclusive classifications.

[19] page fragments repeated within or across documents

2.4.3.3 Content Fragmentation Wrappers

The traditional and most widely used approach to content fragmentation consists of specialised programs, called wrappers, that identify regions of pages based upon various data of interest [Laender2002a]. Most of these algorithms are generally used for content extraction purposes and for this reason aim specifically at identifying the most important region of pages containing the data targeted [Chang2006]. Many of them also aim at adapting HTML pages for mobile browsing [Anam2011, Ahmadi2008] as well as removing clutter content (navigation bars, advertisement etc.) [Louvan2009a, Marek2007a]. Wrappers rely upon pattern matching procedures (e.g., a form of finite-state machines), based on a relatively straight forward set of extraction rules [Chang2006], inferred empirically from recurring structural characteristics of a selected set of pages. Creating a wrapper for a set of HTML pages, for example, corresponds to inferring a grammar for the HTML code used to create these pages and then use this grammar in order to parse each page and extract pieces of data [Crescenzi2001]. This technique isn't limited to only HTML pages and can be used for any file format. In contrast with DOM feature-based content fragmentation algorithms (presented in section 2.4.3.5), which rely upon the result of some form of feature analysis of pages to be fragmented, wrappers simply rely upon tag pattern observations between pages.

The most straight forward, and less sophisticated, wrappers consist of hard coded hand crafted extraction rules, tailored for a limited number of predefined pages with a common structure and layout; these wrappers are commonly also referred to as screen scraping algorithms [Pappas2012]. They can be written either with traditional programming languages [Laender2002a] or using tools specifically designed for this purpose (such as XWrap [Liu2000a], TSIMMIS [Hammer1997] or Minerva [Crescenzi1998a])

[Kaasinen2000, Lin2002a], for example, consider interpreting the tag information of HTML pages to determine specific regions of pages. Tags corresponding to paragraphs, headers, tables or lists are used to divide pages into equivalent fragments.

Content wrappers can be used to fragment just about any resource, as long as its

structure is available to the designer developing the fragmenter. The accuracy of fragments produced is very high, since each wrapper is produced for a dedicated predictable sets of resources. The drawback however, is that content wrappers are generally over-fitted to these particular resources. Hence, in addition to being very labor-intensive to produce [Lo2006] , this results in very brittle fragmentation algorithms, unable to handle minor changes in pages targeted, and therefore very difficult to maintain [Crescenzi2001]. These algorithms assume that pages are both static and identical with respect to their structure. For these reasons, this type of fragmenter is typically used today either as a rapid short term solution or as a last resort alternative if no other technique could successfully fragment the content targeted with acceptable precision.

So as to overcome these drawbacks, wrapper induction systems have been designed in order to automate the process of learning rules to produce such content fragmentation algorithms [Chang2006]. Given a few example pages, wrapper induction is the problem of learning a wrapper that can then be applied to other pages of similar structure to extract the same set of data [ChakrabartiD2010a]. RoadRunner [Crescenzi2001] for example, uses a matching technique called Align, Collapse under Mismatch, and Extract (ACME). Given a set of two HTML pages, an initial version of the wrapper is derived based upon the tag structure of the first page. The wrapper is then progressively refined by discovering regular expressions common to the two pages. This is done by solving mismatches between the wrapper and the sample pages. Although these wrapper induction algorithms do reduce the time required to produce regular expression matching rules, they nevertheless require a lot of manual effort in both selecting and labelling appropriate pages [ChakrabartiD2010a]. Moreover, tailoring these content fragmentation algorithms to new requirements (depending on the text type, domain, and extraction scenario) is a task that varies in scale [Chang2006].

The hypothesis underlying fragmentation wrappers assumes that multiple pages share common tag patterns. However, the design patterns used to present contents on the WWW are unlimited and thus cannot be covered by a finite set of rules

[Alcic2011]. Due to the flexibility of the HTML syntax, conceptually and visually similar pages can be implemented by completely different HTML codes [Zou2007]. Moreover, the more CSS is used, the less important the semantics of a HTML tag becomes [Kohlschutter2010a]. For this reason, different wrappers are required for different sets of pages, which leads to higher maintenance requirements [Lo2006]. [ChakrabartiD2010a], for example attempts to automate the clustering of pages in order to select an appropriate wrapper for each page. Hence as long as the set of pages targeted for fragmentation is relatively small and predictable over time, content wrappers are a relatively straightforward option for content fragmentation, with very good precision. However, for all the reasons described previously, fragmentation wrappers are instead mostly incorporated as extensions to alternative fragmentation algorithms, in order to improve accuracy performances [Kohlschutter2008a].

2.4.3.4 Vision Based Methods

Vision-based content fragmentation approaches operate by simulating the process of user-perception, in order to distinguish different region of pages [Hu2009, Cai2003a]. They build on the hypothesis that web designers provide visual cues that help people recognise different regions of which a document is composed of, e.g visual separators (lines, vertical and horizontal space), background colour etc. [Sleiman2012]. What distinguishes them from other approaches is that they all rely on the page to fragment being rendered. Pure vision techniques, such as colour scale or Optical Character Recognition (OCR) analysis, are used along with hybrid visual/DOM based approaches.

The top-down BESUS[20] [Antonacopoulos2007a] method for instance initially converts scanned pages in pseudo-greyscale formats using a low-pass adaptive filter. Morphological open and close operations, from the field of computer vision, are then used to convert the greyscale image into a matrix. This page representation provides the ability to distinguish regions of pages containing images or text by analysing each line of the matrix. The hypothesis considered by this approach assumes that text lines exhibit regular band structures whereas images do not. Both different sets of

[20]BESUS stands for Bengal Engineering and Science University, Shibpur

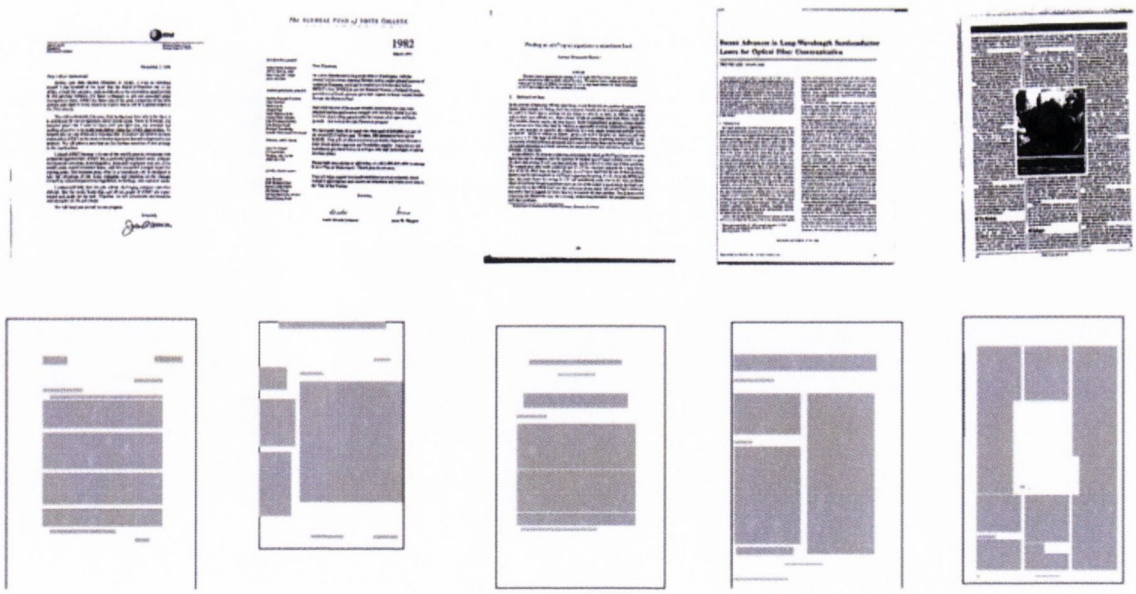


Figure 2.14: Layout Analysis [Hu2000]

bands are therefore detected and grouped together. Using statistical estimations of the height of text bands and line gap, text is then further fragmented into paragraphs. The result of this process consists of fragments containing either images or blocks of text. In contrast to other visual approaches (see below), text regions can be detected in multiple page orientations.

Layout analysis can also be used to compare scanned documents with existing sets, already indexed by a company, in order to extract relevant information for further indexing. The underlying assumption of these techniques is that layouts can determine the kind of content of pages (e.g. research article, news article, letter in figure 2.14) and that business documents usually follow a common template with similar words repeated across documents. Based upon this information, standard template[21] regions of pages can be identified across documents and relevant information contained within regions detected can be extracted.

[Esser2012] more specifically, use OCR analysis for example in order to detect the position of words contained within each page. Each combination of word and position

[21] non visual template detection method will be presented in section 2.4.3.6

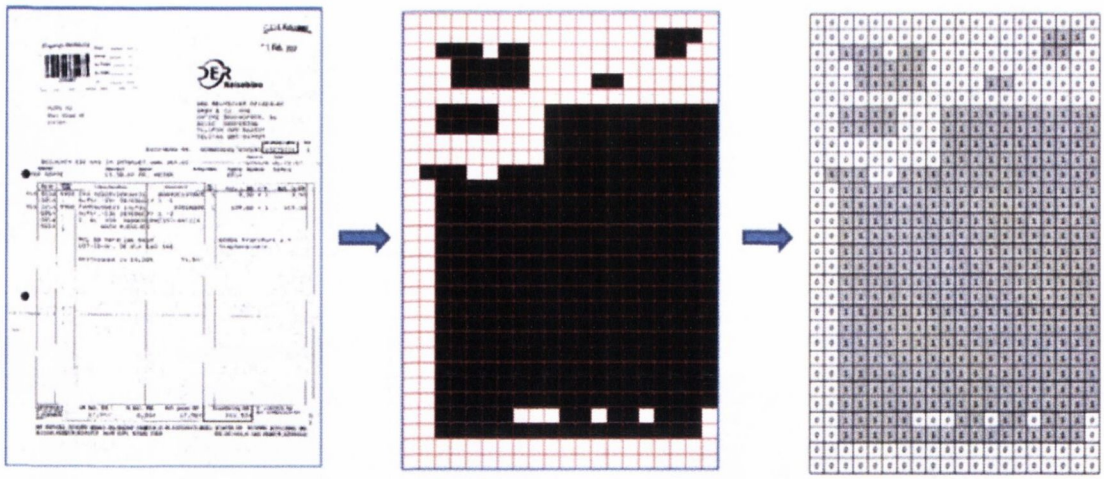


Figure 2.15: Raster Analysis [Esser2012]

is stored in an Apache Lucene index[22] and compared with previous indexed data. If the number of occurrences of a word, in the same position, passes an arbitrary threshold, this word is considered part of a template across documents. Such consecutive words are then grouped into separate regions using a bottom-up approach. Because of its reliance upon OCR, this technique is computationally expensive and requires pages to be processed with the same page orientation. [Hu2000] on the other hand avoid the use of OCR altogether and identify layouts using a template detection method, which converts each document into a binary raster[23] (figure 2.15). The raster can consist of only black (text) or white (background) cells, and sequences of cells are then group together as in the previous approach. Cluster distances between sequences are calculated across documents, and templates are subsequently detected using a sub-optimal path discovery process. This information is then used to identify various regions of pages, in which relevant data is extracted from.

Because approaches described above rely exclusively upon an image representation of pages, any page renderable (including scanned pages) could be theoretically targeted for fragmentation. Nevertheless, most document possess an underlying structure,

[22] http://lucene.apache.org/core/ [Accessed: October 3, 2014]

[23] A raster represents continuous data over space. Raster data is divided into rows and columns, which form a regular grid structure

which could be used to improve the precision of fragmentation procedures. This is what hybrid visual/DOM based approaches [Fumarola2011, Ahmadi2008] attempt to achieve.

The VIsion-based Page Segmentation (VIPS) structural fragmentation approach developed by Microsoft Research Asia [Cai2003a], is probably the most popular example of such a technique. It is been used as a basis for other algorithms extensively throughout the literature [Rao2012, Tan2012, Alassi2012, Song2004]. In contrast to the former approaches, which only analyse a page's rendering in order to detect regions of pages, this algorithm also analyses the HTML visual attributes (font, colour coding etc.) and tags of web pages. It proceeds in two phases, each respectively consisting of a top-down, bottom-up procedures. The DOM tree of the HTML document is initially built and enriched with information regarding visual features such as position, background colour etc. The entire page is initially considered as one large region. Each level of the DOM tree is then traversed and analysed in order to determine whether it could be broken into subregions based upon a set of HTML tag heuristics. Once each subregion is discovered, the algorithm identifies a set of separators (consisting of visual boxes that do not intersect with any of the sub-regions) and assigns each a weight corresponding to the visual differences between the regions it separates. The visual separators are assigned weights based upon their colour, position etc. Consecutive regions of a page, which are separated by a separator with a weight inferior to a predefined threshold, are then merged together through a bottom-up process. The result of these operations consist of a tree of regions and subregions (figure 2.16).

The strength of this approach resides in the fact that it benefits from all the advantages of visual fragmentation algorithms but also takes advantage of hidden clues provided by the HTML syntax as to the underlying structure of each page. Although very popular, hybrid fragmentation techniques such as VIPS however, offer a smaller range of pages that can be processed in comparison with pure visual alternatives, since they require the internal structure of pages to be based upon a HTML like format. Additionally, depending upon decisions made by original author of pages, different

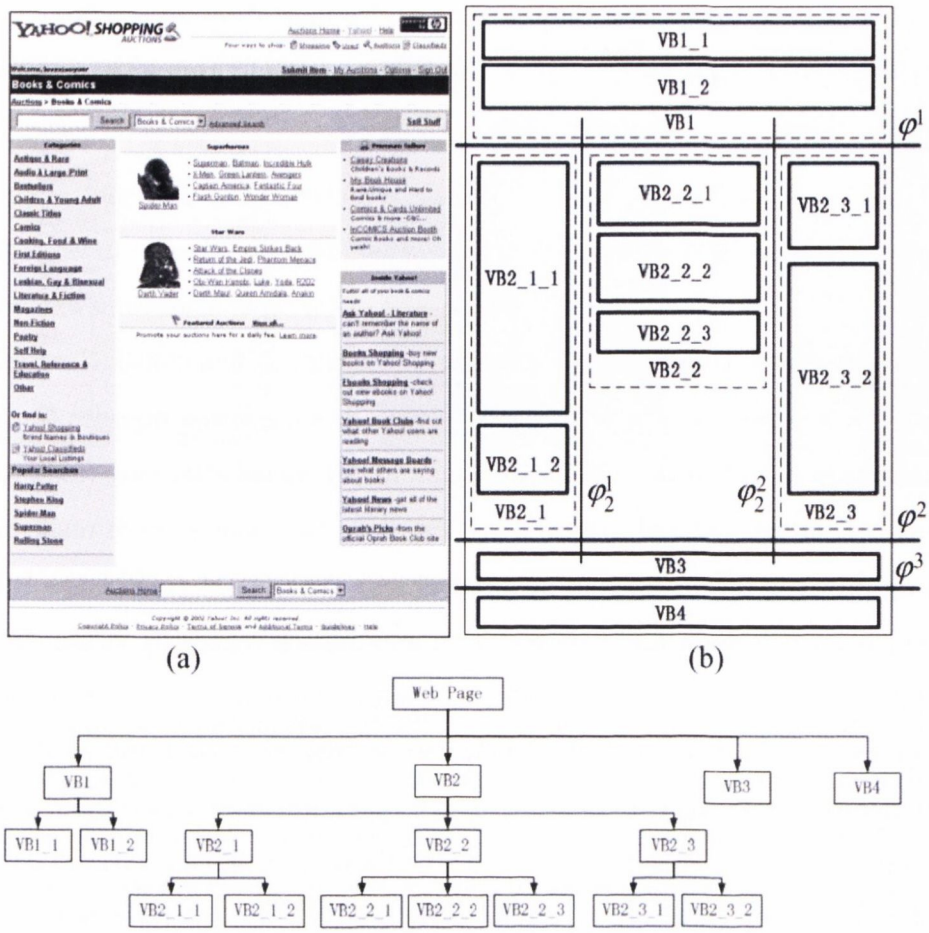


Figure 2.16: VIPs Content Fragmentation [Cai2003a]

HTML elements of a page might be interpreted differently from what was originally intended by this format. For example [Cai2003a] reports very low performances using VIPS for pages using very thin images (representing a line for example) as separators.

2.4.3.5 Feature Analysis Content Fragmentation

Feature analysis methods distinguish themselves from other content fragmentation approaches by relying extensively upon the analysis of internal HTML tag features, in order to determine suitable fragmentations. The vast majority of these algorithms apply their analysis upon HTML web pages, however a large number of these could theoretically also be applied to XML documents if needed [Kohlschutter2008a, Gottron2008a].

Features analysed typically comprise of either content-based features (attributes enclosed within tags such as token length, token frequency etc.) [Kohlschutter2010a] and/or markup based features (attributes that determine the visual representation of content including colour, position, font etc.) [Cao2008]. These fragmentation approaches can be divided into two broad classes depending upon whether they rely on Machine Learning (ML) algorithms or not.

ML based approaches typically use a wide range of features both content and markup based. They require an initial training data set consisting of different regions of pages containing specific tag features and labelled as individual fragments. ML algorithms are then used to determine the most adequate feature combinations characterising fragmentation boundaries between regions. Once these combinations are established, pages processed for fragmentation are analysed, and regions are determined according to the set of features contained in different parts of each page. [Gong2012] and [Marek2007a] for example convert the process of fragmenting pages in to a classification problem. [Gong2012] initially parses the DOM tree of each HTML page in order to produce a set of Basic Objects (BOs). A BO is the smallest information object that can't be divided. It can perform some function only as a whole. Conditional Random Field (CRF) methods are then used to label each of these BOs with a predefined set of labels (information block, navigation block, interaction block and trash block). Consecutive BOs, assigned with identical labels, are finally grouped together into larger blocks in order to form the final regions of each page. [Weninge2008] on the other hand, converts the fragmentation problem into a clustering exercise. This approach transforms each page into a two dimensional histogram using Text-to-Tag Ratios (TTRs) estimates. Clustering algorithms such as Farthest First [Hochbaum1985], K-Means [Macqueen1996] and EM [Dempster1977] are then used to cluster different parts of pages together according to their TTR values.

The use of ML techniques enables content fragmentation processes to rely upon the combination of many features simultaneously in order to identify precisely different regions of pages. However, as they require a training phase to do so, they only

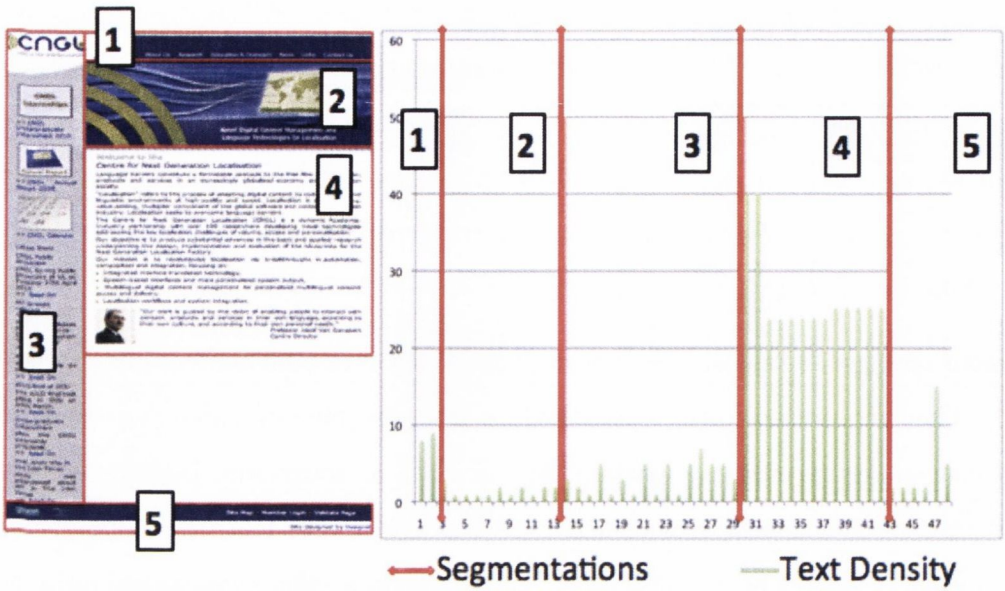


Figure 2.17: Densitometric Page Representation

provide a semi-automatic fragmentation deployment, which could impact scalability. In order to train such algorithms, an initial set of labelled representative training data must be available. This assumes both that a mechanism for selecting these pages is available and additionally that such a truly representative sample does exist for the wide diversity of pages available on the WWW [Spengler2010].

Non-ML approaches, on the other hand, typically rely on fewer features to distinguish differing sections of pages and simpler analysis. These range from the most simple approaches considering only character length of content within tags [Arias2009b], to algorithms generating intermediary data, derived from raw features considered within pages, to identify page regions [Lin2002a]. [Kohlschutter2008a] for example, introduces the notion of text density and assigns a value to all the leaf tags within a page DOM tree. Text density of a tag is defined as the ratio between the number of words and the number of lines within that tag, while a line consist of an arbitrary number of successive characters. The result is a one dimensional representation of each page as depicted in figure 2.17. The fragmentation process thereafter simply consists in identifying differences in text density values above an arbitrary threshold value.

[Gottron2008a] on the other hand computes a Content to Code Ratio (CCR) value based upon the ratio between the number of words to tags encountered within pages and local averages in different parts of each page. A *code blurring* technique, inspired from the field of image processing and using a Gaussian weight distribution, is then applied iteratively to the set of the CCR representation of each page, to alter each CCR value based upon influences of neighbouring values. The process stops whenever CCR values across a page stabilises.

Although the use of pre-determined functions and smaller sets of features can reduce the fragmentation precision of pages, these algorithms achieve performances close to their ML equivalent [Kohlschutter2008a]. Nevertheless, the simplicity of these approaches, in comparison to their ML counterparts, is worth considering with respect to time performances. Moreover, as they are completely automated processes, they do not rely upon any training phase and are legitimate candidates for large scale deployments.

2.4.3.6 Template Detection Content Fragmentation

Among the structural fragmentation approaches reviewed, a group of algorithms distinguishes itself by detecting template portions of documents repeated across pages. Although the increasing use of template structure within web pages (containing navigation links, sidebars etc.) has considerably improved the browsing experience of users (by providing easy-to-access and useful information to users), these templates sidetrack the main topic of discourse in web pages and are now increasingly perceived as "content pollution" [ChakrabartiD2007a]. Their presence, which constitutes close to half of all HTML content on the web [Gibson2005], can affect the performance of search engine indexing, duplicate detection or ranking functions [Bar-Yossef2002a]. Although, these approaches are primarily aimed at detecting duplicate content across pages, they are nevertheless considered content fragmentation techniques since they can distinguish informative from non-informative regions of pages. The template-based content fragmentation approaches reviewed, as part of this research, can be sub-divided into structure and feature detection techniques. Both techniques rely upon an analysis of sample sets of pages, from a single website, prior to their

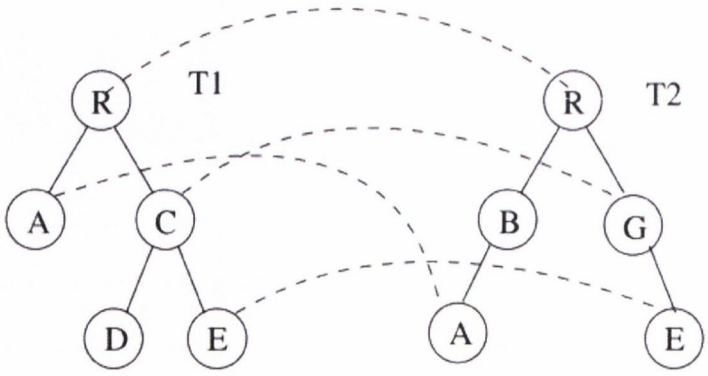


Figure 2.18: Tree Mapping [Karane2006a]

corpus wide content fragmentation operations. Techniques as these, dependent upon a site level analysis, are referred to within this thesis as site level fragmentation techniques.

Structure detection approaches [Bar-Yossef2002a, Karane2006a] identify recurring DOM structure among sets of pages. [Karane2006a] for instance, attempts to discover mappings between the DOM tree structures of pages using the tree-to-tree correction technique introduced by [Tai1979]. This technique computes the cost of transforming a tree into another using edit operations[24] (figure 2.18). Each edit operation is associated with a unit of cost. As many mappings can exist between two trees, the algorithm hence consists in discovering the optimal mapping, which possesses the minimal associated cost. Performing these operations is computationally expensive, hence this analysis is performed only over a small sample set of pages (twelve pages on average). Once a common sub-tree is identified across all sample pages, new pages processed for fragmentation are analysed, using an inexpensive procedure, in order to identify this template sub-tree.

Feature detection techniques [ChakrabartiD2007a, Lo2006] on the other hand, avoid the computationally expensive process of analysing DOM tree structures and instead rely solely upon tag features to detect templates across pages. [Lo2006] for example parses HTML pages in order to extract words contained within tags along with their occurrence paths. Templates present across documents are detected based upon the

[24] vertex removal, vertex insertion, vertex replacement

frequency of word occurrences between page sets. [Debnath2005c] on the other hand, relies upon features such as the number of images, scripts, lists, etc. It converts DOM tags into blocks, which are assigned feature vectors representing the type and number of features contained within each tag. Cosine differences between each block feature vector are calculated. If their similarity reaches an arbitrary threshold, they are considered identical. Inverse block document frequencies are then calculated based upon how often blocks appear across web pages. Blocks which possess high inverse block frequencies are considered template content.

Approaches which combine both structure and feature detection techniques can also be used [Pappas2012, Yi2003a]. [Yi2003] for example proposes to merge DOM trees (union), of web pages which belong to a common website, into an accumulative compressed representation named the Site Style Tree (SST). The SST captures feature (layout, text, images) commonalities of all DOMs analysed on an entire website. Cumulative importances of path of similar nodes are used to compute weights. The higher the frequency of a node, the more likely this node is to be considered part of template material. Although authors of this technique claim this approach is highly effective, it nevertheless requires on average 400 to 500 pages from a single website. This is significant compared to the tree mapping technique described above, which only requires a dozen pages.

Although some methods claim to be page-level structural fragmenters [ChakrabartiD2007a, Debnath2005c] all of these techniques do require some form of site-level analysis to be performed at some stage in their algorithm and are computationally expensive [Buttler2004a]. This assumes that two or more labelled pages from the same domain will always be available in order to apply template detection techniques, which necessarily requires some manual-labor. Moreover, web pages increasingly possess regions that change dynamically (e.g recommended products in Amazon) but can still be considered as noise. Due to their approach, template-based removal methods would fail to detect this type of content [Pappas2012].

2.4.3.7 Graph-Based Content Fragmentation

An emerging set of approaches, based upon graph analysis, attempt to solve the content fragmentation challenge by converting the DOM structure of each page into multiple weighted graphs. Weights assigned to each edge correspond to the cost of placing a node or subtree into the same fragment as another. Once this conversion is performed, the fragmentation process consists in finding an optimal solution, using a large range of standard algorithms, minimising the cost of fragmenting pages across each set of parallel graphs. [ChakrabartiD2008a] for example converts pages into three separate graphs based upon i) stylistic properties (headers, font styles, links), ii) node real-estate and iii) node separation within the original DOM tree. Two algorithms based upon correlation clustering and energy-minimising graph cutting techniques are subsequently used to solve the graph optimisation problem. [Yin2005] converts pages into five separate graphs, and uses random walk algorithms to assign DOM nodes to each fragment. [ChakrabartiD2010a] on the other hand, adopts a more original approach by analysing the relationship between search query logs and pages returned for each query. Specific portions of pages are assigned weights based upon whether the words they contain repeatedly match individual keywords found in search queries. This set of algorithms converts the process of fragmenting pages into a graph optimisation problem, and hence benefit from the research output of this field which has been extensively studied over the years. Nevertheless, as this approach heavily relies upon multiple structural analysis of individual pages, they are necessarily computationally expensive [Karane2006a] in comparison to other methods presented in this review.

2.4.3.8 Semantic-Based Fragmentation

Another set of algorithms, aiming at fragmenting web pages, rely upon the semantic analysis of concepts contained within pages. [Leoncini2012] for example extracts all textual content of a page and uses Natural Language Processing (NLP) algorithms to identify individual words and sentences. Concepts (figure 2.19) are subsequently

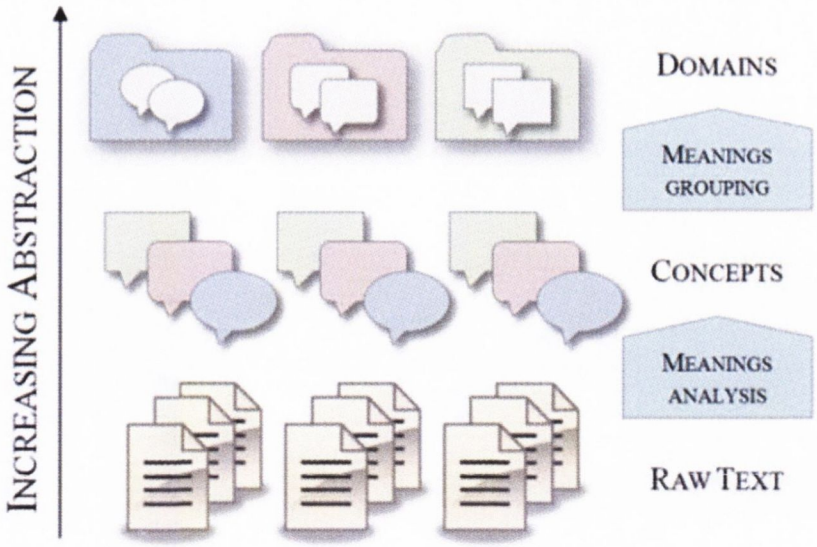


Figure 2.19: Abstraction Layers [Leoncini2012]

assigned to each word, using EuroWordNet synsets[25]. The most popular concepts within each sentence are then grouped in homogenous sets, and for each of those concepts, a list of domains is extracted based upon domain labels available within WordNet[26]. A domain within WordNet is a structure that gathers different synsets belonging to a common area of interest. Similarly as for concepts, the most popular domains are extracted. Sentences are finally assigned weights based upon whether the concepts assigned to its tokens pertain to one of those most popular domains. Each page is finally fragmented based upon these final domain topics and weights assigned to each sentence.

[Mehta2005] on the other hand, initially uses the VIPS algorithm (section 2.4.3.4) to obtain an initial fragment tree of a page. VIPS leaf fragments are then analysed using a naive Bayes classifier and assigned to ODP categories. If a leaf fragment is assigned to more than one ODP category, it recursively runs the VIPS algorithm on this fragment until new sub-leaf fragments can only be assigned one ODP category. Resulting single-ODP leaf fragments are then merged together into larger fragments if they were assigned the same ODP category. The result of this process is a tree of

[25]http://www.illc.uva.nl/EuroWordNet/

[26]http://wordnet.princeton.edu/

single ODP category fragments. The content of each leaf fragment in this tree may possibly be distributed across the page.

Semantic based algorithms, offer the ability to fragment resources based upon the structure of pages taking into account as well the semantics contained within individual structural units of pages which is a unique proposition across the algorithms reviewed for this research. However, their reliance upon individual words and concepts means they are necessarily linguistically dependent.

2.4.3.9 Summary and Discussion of Fragmentation Approaches

As pointed out in section 2.4.3, due to the growing necessity to distinguish individual regions of pages, many content fragmentation approaches have been developed over the years to achieve this purpose. A comparison of the surveyed techniques can be found in table 2.1 and 2.2.

Chapter 1 described how the requirements by AHSs for greater volumes of content is driving the need to process vast amounts of diverse resources accessible via the WWW. For this reason, the approaches presented in this review were compared in priority with respect to characteristics supporting scalability. The characteristics considered consist of: i) semantic/linguistic dependency ii) time performance iii) supported content format(s), iv) targetable range of pages, v) adjustability and vi) sizing.

For the purpose of this review, ***fragmentation scope*** refers to the total set of pages targetable by a method over any time (see figure 2.20). The fragmentation scope of an approach is decomposed into three sub properties consisting in the a) supported content format(s), the b) targetable range of pages and c) adjustability. While the content format property refers to the type of pages which an approach can target for fragmentation, the targetable range refers to the collection of pages belonging to the supported format, which can be fragmented for a given algorithm configuration. Adjustability refers to the degree of effort involved in moving from one target range of pages to another. Given two approaches supporting the same set of content format(s), high fragmentation scope would refer to an approach with either a targetable range of pages equal to the set of resources belonging to the supported formats, or a lower

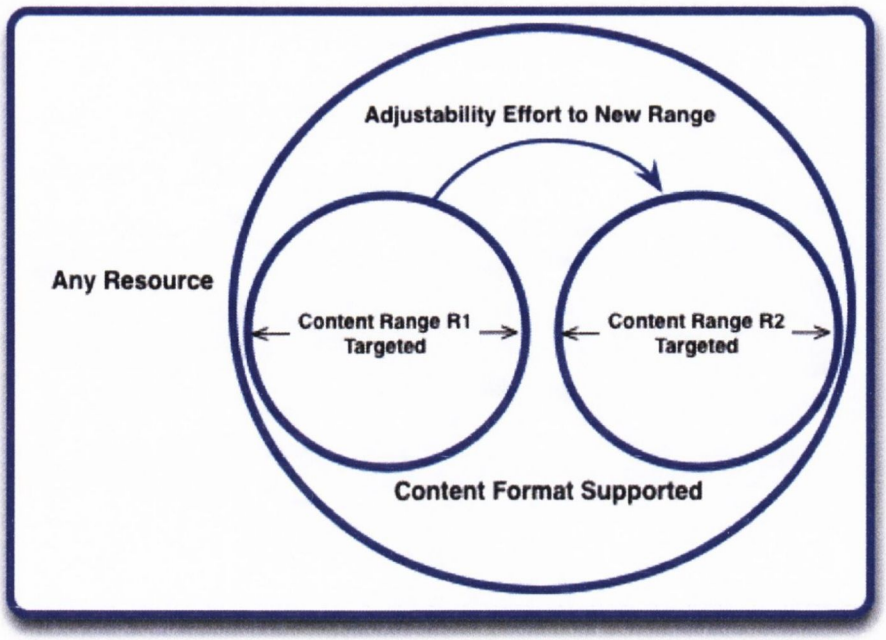


Figure 2.20: Fragmentation Scope

targetable range with an algorithm easily adjustable from one target set of pages to another. Inversely, an approach with low fragmentation scope would possess a small targetable range of pages and high level of effort required to adjust the algorithm from one target range of pages to another.

With respect to ***content formats***, pure visual methods outperform all the other solutions reviewed since they can fragment any digitally rendered document [Burget2007]. This includes web pages but also scanned images of documents [Esser2012, ICDAR2011] as well as Portable Device Formats (PDFs) [Hassan2005]. The vast majority of other techniques, on the other hand, focus upon HTML web pages with many of them theoretically applicable to XML documents too. Nevertheless, most of these algorithms have mostly been evaluated only over a specific subset of such content (news, blogs etc.) [Gupta2005b].

In relation to the ***targetable range*** of pages, visual based, non-ML and graph-based approaches can target any page (belonging to supported formats) for fragmentation, without the need to modify the configuration of selected algorithms. Semantic approaches restrict the subset of targetable resources to a range consisting of pages in

an arbitrary set of supported languages, while template based and ML feature-based approaches further limit their range to resources that belong respectively to a particular website [ChakrabartiD2007a] or similar to an arbitrary training set of pages. The most restrictive approach consists of wrapper content fragmenters. These require the range of pages targeted for fragmentation to be almost identical to a hand-selected set of resources. As soon as the structure of targeted pages changes, wrappers need to be readjusted [Gibson2007a], which makes the approach both very inflexible [Pappas2012] and brittle [Crescenzi2001].

Apart from a few exceptions [Esser2012], once configured, all fragmentation algorithms reviewed for this research are fully automated processes. For this reason, the level of automation indicated in table 2.1 and 2.2 refers instead to the process required to ***adjust*** algorithms from one target range of pages to another. Since visual-based, graph-based and non ML feature-based approaches possess a targetable range of pages equivalent to the number of resources in their supported formats, these techniques do not require algorithms to be adjusted. Semantic approaches on the other hand require the selection of a domain concept mapping services[27] relevant to the new targeted languages, while template based and ML feature-based techniques require new training sets of resources (corresponding to the targeted range of pages) to be assembled as well as labelled (for ML approaches) [Spengler2010] which can be very time consuming [ChakrabartiD2007a]. Once more, fragmentation wrappers require the most effort when it comes to adjusting to a new range of resources. In addition to the necessity for example pages [ChakrabartiD2010a], the process of writing and maintaining wrappers is known to be a very labor-intensive task [Lo2006, Laender2002a, Crescenzi2001] making it very unscalable [Ramaswamy2004]. When data extraction is required from multiple public websites, which frequently modify the format of their pages, the level of effort required to adjust each wrapper to targeted websites makes their use very impractical [Irmak2006].

Hence, among all reviewed approaches, vision-based followed by non ML feature-based and graph-based techniques appear to provide the highest fragmentation scope overall,

[27] For example: a WordNet like equivalent for the targeted language.

while wrappers present the worse.

The techniques reviewed for this research, very rarely considered ***multilingual or semantic dependencies*** of content fragmentation methods. Nevertheless, based upon the description of these algorithms, most approaches would appear to be semantically and linguistically independent. Semantic-based fragmentations, of course represents the exception. Since these techniques rely upon external resources[28] to identify concepts or domain categories assigned to each word within a page, the resources selected to perform these tasks must support both the language and domains of these words. Selected algorithms belonging to the feature analysis [Kohlschutter2008a], pure visual [Esser2012] or graph-based approaches [ChakrabartiD2010a] do rely upon the identification of individual words in pages. Nevertheless, these techniques do not depend upon words pertaining to a predetermined language or domain vocabulary in order to operate. For this reason, they do not portray a semantic or linguistic dependency in contrast with semantic-based approaches. Nonetheless, in the event multilingual fragmentation is required, prior evaluation of a chosen approach should be performed to confirm their semantic or linguistic independence.

With respect to ***time performance***, the estimated computational costs for each approach, as well as the total number of pages to be processed for each page targeted for content fragmentation, were considered the most important factors influencing time performance. Based upon these two criteria, content wrappers and non-ML feature-based approaches appear to be the best options available with respect to time performance. Both approaches rely upon relatively simple computing operations in order to execute (pattern matching for wrappers and simple analysis for feature based approaches [Kohlschutter2010a]) and only require the page targeted for fragmentation to be processed. All other approaches on the other hand rely upon intense computational tasks. OCR techniques and page rendering processes, used by vision-based techniques for instance can be relatively time consuming to perform [Hu2000, Kohlschutter2010a]. ML operations performed by some feature-based

[28]For example: EuroWordNet, WordNet or ODP categories

techniques, as well as the computational complexity involved in structural DOM analysis performed by both graph-based and some template based techniques, can rapidly become overwhelming [Karane2006a, Buttler2004a]. To make matters worse, template based as well as some graph-based approaches [ChakrabartiD2010a] also require many pages to be processed for each targeted page fragmented, which further reduces performance. Finally, the semantic-based techniques presented as part of this review additionally require the analysis of multiple concepts and domains associated with each word within every page fragmented. This would indicate relatively intense computational costs compared to both rule matching and feature analysis mentioned above.

As pointed out by [Kohlschutter2008a], there is no such thing as *the* fragmentation, since correct fragmentations may be considered at different sizes. Nevertheless, the ***sizing of fragment*** outputs is rarely discussed within the literature. There are some exceptions however, most of them within visual-based content fragmentation approaches [Kao2005b, Mehta2005]. With the exception of content wrappers, all sizing techniques identified within this review involve the modification of some predetermined threshold value corresponding to the desired sizing output of fragments. VIPS fragmentation (section 2.4.3.4) for example uses a Permitted Degree of Coherence (PDoC) threshold value to control the size of fragments. The smaller the PDoC, the coarser the output content. Wrappers can also control the size of fragments depending upon the pattern matching rules determined in advance of fragmentation [Louvan2009a]. The literature on this topic remains very vague with respect to the other approaches. Although [Cai2004] claims it is difficult to obtain appropriate granularities for DOM feature-based analysis, [Lin2002a] does affirm pursuing granularity refinement, but does not go into details. In the event the sizing of fragments constitutes an important requirement, further evaluations should be performed upon the selected fragmentation approaches.

	Visual Based		Graph Based	DOM Feature Analysis	
	Pure Visual	**Visual/DOM Hybrid**		**Non ML**	**ML**
Format	Any Renderable Page	HTML			
Range	Any Page				Pages similar to training set
Adjustability	No adjustments necessary				Semi-automated Training required upon labeled data
Time Performance	Low due to rendering process		Low due to structural analysis	High	Low due to machine learning analysis
Semantic and Linguistic Dep.	Theoretically Independent				
Sizing Support	Threshold Modification		Undertermined		

Table 2.1: Content Fragmentation Comparison

	Wrappers		Template Detection	Semantic
	Manual	Induced		
Format	HTML			
Range	Selected type of pages Identified prior to fragmentation		any page belonging to website targeted a priori to fragmentation	pages in selected set of languages
Adjustability	manual	fully-automated	semi automated	manual selection of a concept mapping domain in new targeted language
Time Performance	High		Low due to structural analysis and number of pages to be processed for teach targeted pages	Low (concept and domain mapping required for each word in pages)
Semantic and Linguistic Dep.	Theoretically Independent			Dependent
Sizing Support	Tag Pattern Matching		Undertermined	

Table 2.2: Content Fragmentation Comparison

2.4.4 NLP Algorithms

2.4.4.1 Introduction

The previous section presented content analysis approaches aiming specifically at analysing the structural aspect of resources. However, since the advent of the 1959 Georgetown-IBM experiment [Hutchins2005], involving the translation of Russian sentences into English, a vast amount of research has also been carried out upon the linguistic analysis aspects of documents. Approaches drawn from this field of research are usually referred to as NLP algorithms and use a combination of machine learning and statistical analysis techniques. These algorithms are widely used and now also available as web-services[29].

As there exists a vast variety of NLP algorithms, this section intends to present the reader with an overall understanding of the fundamental approaches and concepts involved within this area of research. Thus, standard techniques used within this field, as well as those most relevant for the rest of this document are presented in this section. The aim in particular, is to serve as a basis for decisions described within the design and implementation chapter of this thesis.

2.4.4.2 NLP Overview

The fundamental objective sought by NLP research is to convert a piece of text into a data structure that unambiguously and completely describes the meaning of the natural language text [Collobert2011]. While this ideal has still to be accomplished, existing NLP techniques are currently used to solve more restricted problems and have been applied successfully in a wide range of areas such as machine translation [Magdy2011], dialogue systems [Skantze2010], Information Extraction (IE) [Fader2011] or IR (section 2.3.5).

To achieve this end, NLP systems split practical problems into a series of consecutive tasks. Each of these tasks represent a research field of its own, and attempts to solve a

[29] AlchemyAPI http://www.alchemyapi.com/, OpenCalais http://www.opencalais.com/ [Accessed: October 3, 2014]

specific objective with standardised metrics and benchmarks in multiple languages[30]. These tasks are executed together sequentially, with the output of one becoming the input to the next, in a so-called NLP pipeline. Since various algorithms can be used to achieve a specific task, this separation of concern into a pipeline enables different implementations to be switched with each other. NLP tasks are usually subdivided into two broad classes based upon whether they consist of low-level syntactic or more high-level semantic tasks. Most NLP pipelines usually initiate analysis using syntactic tasks first, used as a foundation for subsequent semantic tasks.

2.4.4.3 Low level NLP Tasks

The most common syntactic tasks used in standard NLP pipelines consist of tokenization, sentence splitting, Part-Of-Speech (POS), chunking, stemming and lemmatisation.

Tokenization and ***sentence splitting*** tasks attempt to separate a stream of text into a consecutive set of tokens or sentences respectively (figure 2.21). Tokens are often loosely referred to as terms or words, however they can also consist of symbols or phrases. The tokenization task is generally the first performed in an NLP pipeline and serves as a foundation for all other subsequent algorithms. For most languages based on the Latin alphabet, these tasks are fairly straightforward. Although some consideration for exceptions such as hyphenated words needs to be taken into account, in most cases, each token can roughly be defined as a sequence of characters positioned between two white spaces, while punctuation can easily distinguish between two separate sentences. However, for other languages such as Chinese for example, tokenization tasks for example can become quite complex since there exist no delimiters to indicate words separation [Zhang2010a]. A large field of research is hence dedicated to this task alone. [Xue2003] for example, uses the local context of characters along with a maximum entropy model to achieve this task, while [Peng2004] and [Gao2005] respectively use conditional random field and perceptron models to achieve the same objective.

[30]Eacg CoNLL conference for example provides common data sets to conference participants http://ifarm.nl/signll/conll/ [Accessed: October 3, 2014]

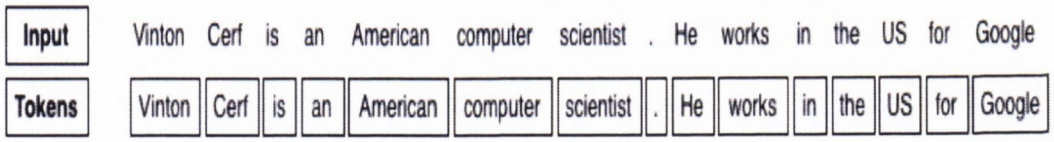


Figure 2.21: Tokenization Example

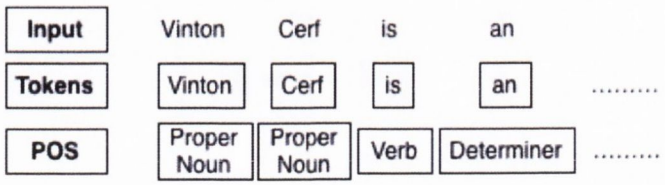


Figure 2.22: POS Example

Once a text has been tokenized, ***POS algorithms*** are usually executed over the resulting set of tokens and sentences (figure 2.22). These algorithms aim at labelling each token with a unique tag that indicates its syntactic role (such as plural noun, adverb etc.) [Collobert2011]. [Ptaszynski2012] for example, use a Hidden Markov Model as a basis for a POS tagger implementation targeted at the endangered Ainu language, while [Shen2007] propose a sequence classification approach for the English language, which they claim obtains an error rate of only 2.67% on standard benchmarks.

While some pipelines might only require text to be tokenised, most fields relying upon NLP techniques (such as IR for example) require various tokens to be grouped together based upon their lemma or stem. A lemma is the canonical form of a set of words. For example, words such as, *run*, *runs*, *ran* and *running* all possess the same lemma consisting of *run*. On the other hand, the stem of a word refers to the root of a word to which an affix can be appended. For example, the words *wait* (imperative), *waits* (present), *waited* (simple past) all possess the same stem consisting of *wait*. The process of lemmatization or stemming is performed through a ***morphological analysis***. Morphological analysis algorithms have been developed since the 1980's and can achieve high performances. These algorithms consist of rules, defined based upon the grammar of a targeted language. The Porter stemming

algorithm [Porter1980] for the English language for example, represents one of the earliest attempts to achieve this task and is today still widely used today. The morphology identification task however can be much more difficult with respect to other languages such as Hebrew and Arabic, due to the higher word structure complexity of these languages compared to English [Jamilah2010].

Finally, chunking, also referred to as ***shallow parsing***, is a task which aims at detecting a series of tokens forming a noun or verb phrase within a sentence. Series of tokens such as *"is going to investigate"* or *"is investigating"* constitute verb phrases in the English language. As for the POS task, shallow parsing can be performed using many approaches going from simple rule-based algorithms [Aduriz2013] to methods using conditional random fields [Sha2003] or hidden markov models [Freitag2000a].

2.4.4.4 High Level NLP Tasks

Among higher level tasks, IE algorithms represent common NLP tasks included within analysis pipelines. Also two other tasks, of particular relevance for this thesis and described below, consist of boilerplate detection algorithms and topical algorithms.

IE algorithms aim in general to extract semantic information from text. Among a wide range of tasks available within this category of algorithms, named entity recognition, relationship extraction and coreference resolution tasks are the most frequently referred to. Named Entity Recognition (NER) [Nadeau2007] aims to detect entities referred to within a text and assign to each a type (such as person, organisation, place name etc.) and unique identifier (figure 2.23). Many machine learning algorithms have been conceived to perform this task. Hidden Markov Models (HMMs) [Bikel1997] or conditional random field [Mccallum2003] techniques for example have been used to provide supervised learning solutions to this task, while unsupervised approaches, that rely upon the use of lexical resources such as WordNet[31], have also been explored [Alfonseca2002]. [Kobilarov2009a] for example, describe a weight calculation technique based upon the PageRank [Brin1998]

[31] wordnet.princeton.edu/? [Accessed: October 3, 2014]

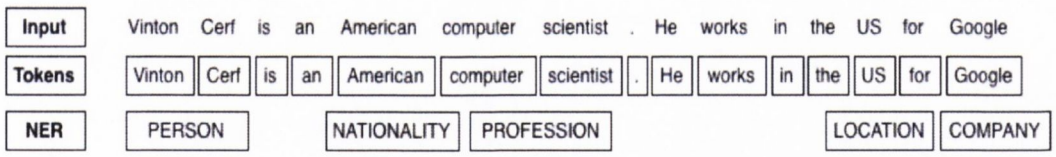


Figure 2.23: NER example

algorithm used by the BBC[32] to discover and assign entities with corresponding DBpedia URIs as identificators.

Coreference resolution [Stoyanov2009] on the other hand, attempts to discover links between extracted named entities. These links can consist of anaphoric or coreference associations. In the text depicted in figure 2.23 for example, "he" and "Vinton Cerf" refer to the same real world entity. Most coreference resolution approaches perform this task by aggregating local decisions about pairs of mentions discovered during the analysis of large collections of labeled data [Raghunathan2010]. Following an NER and coreference analysis, relationships between each entity discovered can be identified. Figure 2.24 for example describes how "person", "organisation" and "location" entities discovered by previous tasks, can be associated with each other depending upon their occurrence within the text. As for previous tasks, many machine learning techniques have been developed in order to achieve this purpose [Ramakrishnan2008]. The general approach involves identifying token morphological variants (such as "work in", "live in" etc.) depicting relationships between entities within a chosen domain. Whenever such morphological variants are identified, the associated entities are grouped based upon this relationship. As manually labeling morphological variants is time consuming, unsupervised methods attempt to discover these variants across large sets of unlabelled resources.

Boilerplate detection algorithms on the other hand [Pasternack2009a, Weninger2008, Kohlschutter2010a], attempt to separate the main content of interest in a page from other elements surrounding it. This NLP task can be used, in similar ways as simple stop-word removal techniques, to improve web search, mining,

[32]http://www.bbc.com/ [Accessed: October 3, 2014]

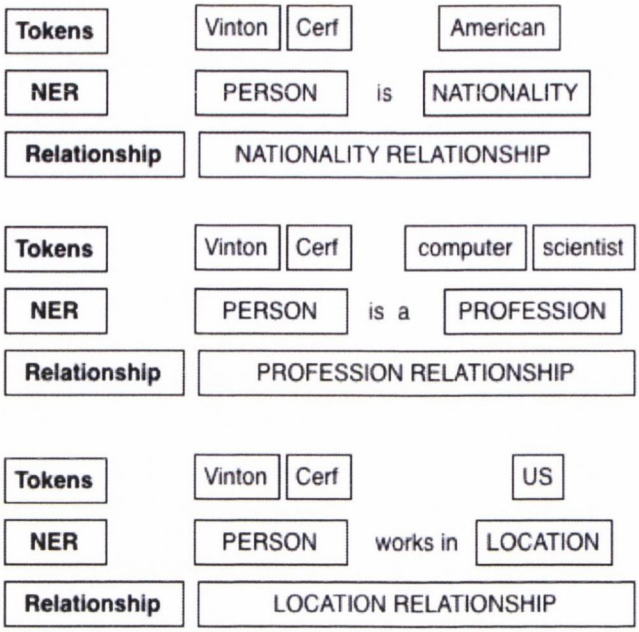


Figure 2.24: Relationship extraction example

or storage capacity [Kohlschutter2010a]. Different definitions of what constitutes boilerplate content have been proposed based upon the application area they are been used in. It is usually vaguely referred to as non-informative parts outside of the main content of a page, repeated between web pages and typically machine generated [Pomikalek2011]. The CleanEval competition [Baroni2008], which is the most popular benchmark used for this task, identifies for example boilerplate content specifically as consisting of navigation, list of links, copyright notices, advertisement, forms, duplicate material etc. Some of the research carried out with respect to content fragmentation techniques overlaps with this field. Template fragmentation approaches for example (section 2.4.3.6) could be considered a form of boilerplate detection algorithm since they effectively aim to separate pages into fragments based upon the detection of sections repeated across a website. [Gibson2005] for example, detect templates within pages, shared across a website by extracting all subtrees of each DOM tree parsed and computing a hash for each one. Occurrences of individual hashes within a website are counted and DOM sub-trees assigned to hashes associated with a value above an arbitrary threshold are marked as template content (i.e. boilerplate). Another approach developed by [Kohlschutter2010a], which does

not rely upon the analysis of multiple pages, detects boilerplate content of web pages by analysing the number of words and link density contained within each tag in a web page.

Finally, ***topical algorithms*** attempt to discover and model the topics covered by resources analysed. The most common encountered consists of topic labelling, topic segmentation and topic classification algorithms. As its name suggests, topic classification algorithms attempt to categorise resources into different categories based upon the topics covered within resources analysed. Most algorithms used to accomplish this task represent documents as bags of words [Boulis2005] and subsequently use different machine learning algorithms to learn a function, which can map individual documents to different topic categories. Whenever labelled document-topic relations are unavailable, unsupervised algorithms, using clustering techniques for example [Kyriakopoulou2008], have been developed to identify similarities between documents. Although most of the research related to topic classification has in earlier years focused upon the classification of documents, the topical classification of social media content is the subject of increasing investigation [Prasad2007]. [Cano2013] for example, use graphs structure analysis of linked data repositories (such as DBpedia or Freebase) to classify tweets, enriched with the results of a prior NER analysis. Each entity discovered is assigned to various linked data concepts and contextual information about each entity, along with their relationship to other linked data entities is used to support this classification.

While approaches presented previously, analyse the topical properties of each resource as a whole, topic segmentation algorithms [Jameel2013, Blei2001, Hearst1997] on the other hand aim to discover individual topics and their boundaries within a resource. A news broadcast for example, which covers four different stories, clearly divides naturally into four different topics. Less clearly, a multi-page magazine article, while overall covering a single broad topic, will usually cover a series of subtopics as it examines different aspects of its subject matter and explores the subject area [Ballantine2004]. The most widely cited and pioneer approach to topic segmentation consists of the TextTiling method proposed by [Hearst1997]. This approach assumes

different topics use different vocabulary sets. As a result this technique locates topic changes within articles based upon statistical word-frequency changes within a document. It uses two overlapping sliding windows of an arbitrary size of 20 tokens each. Each window is modelled using a keyword-vector with values corresponding to the frequency of each word in the window. Cosine similarity measurements are then used to compare both vectors and determine shifts in topics. This simple technique has been used as a basis for many other algorithms [Ballantine2004]. However, it only generates one level topical segmentations. More recent techniques [Song2011b], can produce topical tree structure from resources analysed in order to take into account subtopics discovered within a document.

While topic segmentation algorithms [Lau2010] attempt to discover topic boundaries within text, topic labelling algorithms on the other hand aim to assign descriptive labels that best describes the topic covered by a text fragment. [Mei2007] for example, present a technique which extracts noun chunks from documents using shallow parsing (see previous section), as potential labels, and then ranks these based on their Kullback-Leibler[33] divergence with respect to a given topic (modelled using a range of modelling techniques). [Hulpu2013] on the other hand identify the most adequate label based upon the assumption that words co-occurring in text likely refer to concepts that belong closely together in the DBpedia graph. Graph centrality measures are therefore used to identify which concept represents best a given topic.

2.4.4.5 Gate Overview

Through the implementation carried out for this research, a key technology used to support the incorporation of NLP algorithms within the slicer consisted in the General Architecture for Text Engineering (GATE) toolset. This platform is referred on several occasions within the implementation chapter of this thesis. An overview of this technology is therefore provided within this section. Reasons why the GATE toolset was chosen with respect to other existing alternatives are presented subsequently in

[33]Kullback-Leibler measures the difference between two probability distributions P and Q. The divergence between these two probabilities is a measure of the information lost when Q is used to approximate P

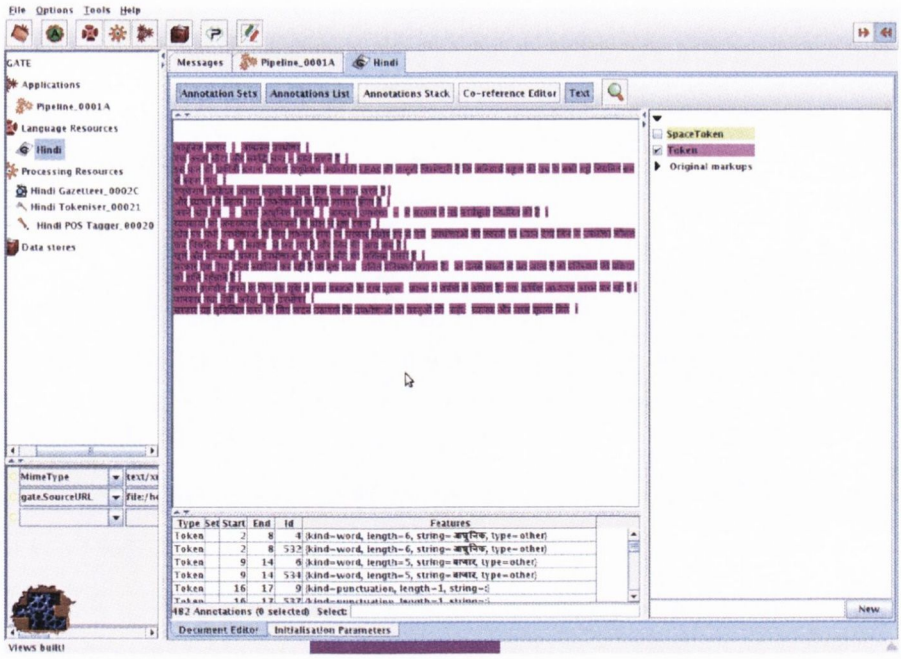


Figure 2.25: GATE Developer

section 4.3.3.3.

The GATE toolset refers to the various tools which have been created over the years. Among these, the two most popular, which consist of GATE Developer and GATE Embedded API, were used to implement the prototype slicer used for this research. The latter consists of a set of Java libraries, which can be included in any Java-based applications in order to provide a flexible pipeline architecture for NLP analysis. GATE Developer (figure 2.25[34]) on the other hand, provides a visual interface to GATE Embedded. This tool was used within this research primarily for pipeline testing purposes, with limited sets of native open-corpus resources prior to inclusion within the overall Slicepedia pipeline. For this reason, references to GATE within the rest of this document refers specifically to GATE Embedded as opposed to GATE Developer.

A GATE pipeline is composed of four major building blocks known as Language Resources (LRs), Processing Resources (PRs), Visual Resources and Controllers. LRs consist of entities such as lexicons, corpora or ontologies. Within the context of

[34]Courtesy of http://gate.ac.uk/

this Slicepedia implementation, LRs simply refer to resources targeted for analysis, which either consist of native open-corpus resources or foundation fragments. PRs refer to primarily algorithmic entities, such as tokenizers, entity extractors etc... while visual resources refer to components used primarily within GATE Developer. As mentioned previously, the rest of this thesis ignores the latter, for this reason visual resource components are also ignored. Finally, controllers simply refer to the NLP pipeline used to process LRs by combining various PRs together.

A family of language analysis PRs, called A Nearly-New Information Extraction System (ANNIE), is provided by GATE. ANNIE includes various common NLP tasks such as tokenisation, semantic tagging or verb phrase chunking. Within GATE, each PR produces a set of annotations representing the output of the task fulfilled by the algorithm contained within that PR. The annotation format used by GATE consists of a modied form of the Grishman1997 format [Grishman1997] originally created by Defense Advanced Research Projects Agency (DARPA). They consist of labels assigned to either a specific range of characters or an entire document. Annotations can be grouped within categories and can be used by subsequent PRs to pursue their task. Annotations can also hold a set of features such as Strings or Integers. GATE supports the incorporation of a variety of document formats[35]. Prior to analysis, each document is converted into a single unied model of annotation. When incorporating a document in HTML format for example, all tags encountered within the documents are converted to annotations of the same name. As a result, each PR within GATE, is executed over the text contained within these tags only. Additionally, GATE also provides the ability to write custom regular-expression based pattern/action rules referred to as Java Annotation Patterns Engine (JAPE) transducers. JAPE rules identify specific annotations produced by a pipeline and, following an analysis of these, can either produce or delete specific annotations as required.

Finally, GATE offers the ability to choose between different controllers. The two NLP pipelines used within Slicepedia consist of real-time controllers. In addition to the standard GATE controller, which simply processes LRs by executing each

[35]Including XML, RTF, email, HTML, SGML and plain text

PR successively, these controllers were specifically designed to guarantee particular response time quality of service when executed within web-services. Whenever a web-service executes a real-time controller over a given set of LRs provided by service clients, this controller will skip the analysis of a particular LR if it detects a PR is requiring an unusual higher amount of time to execute. This prevents web-services from hanging, whenever an LR is corrupted. Controllers within GATE can contain other controllers within each other. This feature is very useful when a set of PRs needs to be executed multiple times for a specific set of fragments within a resource (as described in section 4.2). Lastly, *conditional controllers* are also available if necessary. These controllers permit a succession of PRs to branch depending upon features assigned to documents. Such controllers would be necessary in order to support multilingual slicing (section 4.2).

2.4.4.6 Summary

This section presented the reader with a short overview of NLP algorithms including the most common low level tasks typically involved in this area, as well as higher level semantic analysis techniques. These techniques could clearly be of benefit to the AHS community, with respect to automatically identifying relevant attributes of open corpus resources for various reuse purposes. The design chapter of this document will attempt to provide an example of such a contribution to the community.

2.4.5 Content Reuse and Analysis Summary

Section 2.4.2 presented various approaches, which have been previously successful in improving the reuse of resources within various scenarios. This was followed with the description of multiple algorithms, which have been developed over the years, with the aim of analysing open corpus resources and modifying/extracting various attributes of these resources, so as to improve their reuse for various purposes. In particular, section 2.4.3, presented a review of content fragmentation algorithms aimed at distinguishing various regions of open corpus content based upon their structural aspects of these resources, while section 2.4.4 provided an overview of the range of NLP algorithms now available to the community to perform deeper syntactic and semantic analysis of these resources.

All of these development in recent years, could clearly benefit the AHS community by improving the supply and reuse of open corpus resources within these systems. The next sections of this thesis will therefore focus upon presenting a new approach, addressing the limitations of existing content reuse approaches presented earlier by using new opportunities arising from new content reuse and analysis techniques presented within this section

2.5 Conclusion

This chapter presented the reader with an overview of AHSs, their anatomy and in particular their dependencies upon the content reused for adaptation purposes. These systems require content to be reused to match specific niche content requirements, in order to apply AHS adaptation techniques to deliver personalised experiences to individual users. AHS content models, used to produce and supply these resources, were also presented along with their strengths and weaknesses with respect to the widespread adoption of these systems.

Previously successful forms of content reuse, as well as NLP techniques available for the analysis of open corpus resources presented in this chapter, could form a basis for the development of new content models leveraging opportunities provided by these techniques. The next chapter hence outlines the design requirements of such a content supply system, to be developed as part of this research, based upon influences derived from this chapter.

Chapter 3

Design

3.1 Introduction

This chapter aims to identify and represent the core fundamental design requirements of a service which provides the automated harvesting, customisation and delivery of content packages from open-corpus resources, in order to support the content supply needs of various individual AHSs. The design of this prototype service is focused upon contributing to the second objective of this thesis.

The chapter begins by detailing influences and lessons learnt, emerging from the analysis presented in the state-of-the-art chapter, giving rise to key principles underlying the design of this service. Key principles are then combined together and refined as high level design requirements for a prototypal service performing the harvesting, customisation and delivery of open-corpus material for AHSs. These high level requirements are further refined and supplemented into a set of more specific technical requirements. The proposed service architecture is then presented, followed by the description of two content fragmentation algorithms, used extensively within this prototype. This chapter concludes by summarising the key points discussed in each of the preceding sections. All design decisions and descriptions presented in this chapter were performed by the author of this thesis with the exception of the algorithm presented in section 3.4.2. This algorithm was previously available in the literature and serves as a foundation for a variation of it, designed by the author for the purpose of this research, and presented in section 3.6.

3.2 State of the Art Influences

3.2.1 Introduction

The analysis conducted in the previous chapter influenced various aspects of the applied research described in this thesis. The aim of this section hence, is to present the reader with a summary of these influences along with resulting key principles.

3.2.2 State of the Art Influences

Influences derived from the state of the art chapter, are presented and grouped within eight different categories, each highlighting individual cross-cutting perspectives with respect to the limitations and opportunities identified as part of the analysis performed.

These categories presented in the following subsections consist of: 1) External Content Production; 2) Low Cost Automated Content Production; 3) Open Corpus Content Reuse Improvement; 4) Open Corpus Right-Fitting; 5) Content/AHS Agnostic Open-Corpus Reuse; 6) On Demand Niche Right-Fitting; 7) Content Repurposing; and 8) Web of Slices.

1) External Content Production

a) There remains an insufficient quantity of recomposable resources available to AHSs due to their reliance upon traditional closed-garden content production approaches (section 2.3.3)

b) Content encapsulation techniques can improve the reuse of resources between different AHSs (section 2.4.2.2)

c) Whenever an AHS needs to use content produced for third party AHSs, it must comply to the content formats chosen by each of these independent AHSs[1], which acts as an obstacle towards the widespread reuse of content among AHSs (section 2.3.3).

[1] A formal definition of what constitutes an independent AHS within the context of this thesis is available in the glossary

d) AHSs have modified their architecture with respect to content incorporation and can now incorporate content from external sources (section 2.2.2 and 2.3.4)

Key Principle 1.a: Content should be made available to AHSs through the use of a service (to be named a slicer) delivering resources encapsulated as content packages and directly reusable within *various independent* AHSs. The ability to externalise the delivery of content for AHSs as a service could enable AHS designers to reduce their concerns related to content production and instead let them focus upon the adaptation needs of their platform.

Key Principle 1.b: The "AHS/slicer" relationship should be of type "master/slave", with the slicer providing as much control to AHSs as needed over the content delivered. This configuration would for example remove the need for AHSs to comply to various formats and instead enable AHSs to receive content in their format of choice[2].

2) Low Cost and Automated Content Production

a) The production, repurposing and supply of resources recomposable within AHSs is very labor-intensive and expensive (section 2.3.3).

b) The need for content production to meet increasingly more niche content requirements of AHSs, means the demand for larger and more diverse volume of content will increase over time (section 1.1).

Key Principle 2: The production and delivery of content for AHSs should be achieved *fully automatically* and at *low cost*. This would reduce significantly production costs and support an increase in the volume of content available for AHSs.

3) Open Corpus Content Reuse Improvement

[2]The delivery of content in multiple formats is a relatively straight forward engineering task. Hence, although mentioned as part of the Slicepedia design requirements, the rest of this research ignores this requirement in order to focus instead on more novel aspects of this slicer.

a) The web now offers a large quantity of diverse resources on a vast range of topics. (section 2.3.4)

b) There are numerous techniques developed within the IR community to discover, harvest and retrieve relevant open corpus resources (section 2.3.5)

c) As mentioned before in influence 1.*d*), AHSs have started modifying their architecture to incorporate external content.

Key Principle 3: The delivery of content to AHSs by the slicer should be based upon the identification (via standard IR techniques) and *reuse of open corpus* resources. The reuse of open corpus resources within AHSs would dramatically increase the volume of content at their disposal.

4) Open Corpus Right-Fitting

a) Open corpus content can currently only be accessed automatically by search engines as one-size-fits-all document level content packages (section 2.3.4 and 2.3.5).

b) In order to recompose content, AHSs require resources to match specific content requirements. For this reason, document level delivery and incorporation of native open corpus resources is inadequate for AHSs (section 2.3.2).

c) The usage of content specific techniques to incorporate web resources within AHSs only addresses the content needs of few and pre-determined AHSs with specific content requirements, over a limited range of open corpus content (section 2.3.4.3).

d) Many algorithms have been developed to automatically identify the structural units of pages (section 2.4.3) and analyse syntactic/semantic properties of content (section 2.4.4).

Key Principle 4.a: The slicer prototype should provide the correct selection

and right-fitting[3] of native open-corpus resources[4] to support the recomposition[5] of slices by AHSs, through the use of content fragmentation and semantic analysis algorithms. The right-fitting of native resources should lead to Self-Contained and Independent (SCI) content[6] (content packages which are both concise and self-sufficient). The ability to do so could expand the reuse of open corpus content (at various levels of granularity, with only relevant portions of pages delivered and with appropriate meta-data) to many AHSs as opposed to pre-defined consumers.

Key Principle 4.b: Since the process of converting open corpus content into right-fitted content packages relies upon various fields of research (i.e. content fragmentation, semantic analysis etc.) the architecture of the slicer prototype should be *flexible* in order to enable individual components, dependant on each field, to be replaced as each state-of-the-art improves.

5) On Demand Niche Right-Fitting

a) The production of resources is usually performed at design time, a-priori of system deployment or users interacting with the system which involves predicting the demand for content which involves high risks (section 2.3.3).

b) The move towards more niche content requirements means the ability to predict content consumption needs is becoming increasingly difficult (section 1.1).

c) The use of open corpus content entails the necessity for open domain models. The Linked Data revolution now provides the community with a

[3]Throughout this thesis, the process of right-fitting refers to the production of content, resulting in the customisation of preliminary native resources, meeting an arbitrary set of niche content requirements. A more detailed definition is provided in the glossary

[4]Native resources refers to content accessible in a format intended by the original author, in other words resources prior to being sliced.

[5]As part of an initial attempt to evaluate the slicing of open-corpus resources to improve its reuse, it is an approach of this research to minimise the degree of recomposition carried out by the slicer. Instead, any recomposition decisions are delegated to AHSs.

[6]A detailed definition is available in the glossary

large variety of concepts and ontologies (section 2.4.2.3).

Key Principle 5.a: Niche content requirements of AHSs should be provided to a slicer only at run-time, which means that the delivery of content packages by the latter should be performed *on-demand*. This approach would both i) lower risks related to content demand prediction and also ii) provide AHSs with resources matching their content requirements as close as possible.

Key Principle 5.b: In order to introduce content packages (produced through the reuse of open resources) within various independent AHS with domain models unknown at design time, the selection and right-fitting of native resources should at a minimum be perform based upon open domain models. Linked Data concepts could served as anchor points for an *open domain model*, which individual AHS domain models could connect to. This could enable a tighter integration between internal proprietary resources used within AHSs and open resources provided by a slicer.

6) Content/AHS Agnostic Open-Corpus Reuse

a) Open corpus content is very diverse. It comes in different formats, languages and is associated with a lot of non-informative content (such as navigation bars, advertisement etc.). (section 2.3.4).

b) The use of content specific techniques to reuse open corpus resources within AHSs is very labor-intensive. These techniques can only target predefined sets of web resources, for reuse within established AHSs with pre-determined content requirements. This limits the volume of open corpus content which can be leveraged within AHSs (section 2.3.4.3).

c) As mentioned previously in 4.*d*), many algorithms have been developed to automatically identify the structural units of pages and analyse syntactic/semantic properties of content.

Key Principle 6.a: The process of reusing open corpus resources should be performed by the slicer using a *content/AHS agnostic* approach.

Key Principle 6.b: Since content packages produced should be directly reusable within *various independent AHSs* (Key Principle 1.a), the slicer should also support the *search & delivery* of content packages over multiple channels.

The ability to automatically reuse open corpus resources, without knowing in advance what resources are targeted for reuse, could move content production approaches beyond the reuse of only pre-selected resources and leverage the full diversity and volume of open corpus content available on the web.

7) Content Repurposing

a) Up until recently, the reuse of resources was usually performed for predefined purposes determined by the original authors of content (section 2.4.2.4)

b) AHSs rely on page adaptation techniques (link hiding etc.) to provide personalised content experiences to users (section 2.2.3)

c) A variety of NLP analysis tools are now available to analyse various features of open corpus content (such as topic covered, named entity recognition, part of speech etc.). (section 2.4.4)

Key Principle 7: The right-fitting of native resources into reusable content packages should be performed in order to support page adaptation techniques used by AHSs. NLP algorithms could be used to supply AHSs with meta-data supporting the *repurposing of open-corpus content* through page adaptation by AHSs. Reusing existing resources for purposes not intended by original authors could unlock latent potential of published material.

8) Web of Slices

a) Various approaches, that rely upon shared publishing and delivery mechanisms, have been previously successfully elaborated in order to maximise the usage and reuse of existing resources (i.e: the WWW, Linked Data etc.) (section 2.4.2.3).

b) The quality, precision and range of syntactic/semantic analysis algorithms currently available is now very diverse (section 2.4.4)

c) As mentioned in influence 2.*b*), the move towards niche personalisation means the diversity in niche content requirements is increasing.

Key Principle 8.a: In order to support the supply of ever larger volumes of resources to AHSs, the generation of content packages as well as the underlying resources needed to produce such packages by slicers should be *open and shareable* between institutions.

Key Principle 8.b: To promote collaborative efforts, the *low cost* requirement mentioned previously should also apply to the implementation of slicers. Linked Data and RDF technology could be used through a so-called *web of slices*[7] in order to leverage the reuse of content analysis output performed by separate institutions, and hence support a wider range of niche content requirements from AHSs.

3.2.3 Summary

The analysis conducted in chapter 2 revealed, i) limitations with respect to the underlying content production approaches currently used to supply AHSs, as well as ii) new opportunities offered by early signs of open content incorporation techniques and state-of-the-art content analysis tool performances. This section summarised these influences and key principles (listed in table 3.1) made with respect to the prototype service to be developed for this research. The next section provides the reader with a description of high level design requirements, derived from the key principles enunciated in this section, of a service which enables the harvesting, customisation and delivery of content from open corpus sources.

[7] A formal definition of a slice is available in the following section

ID	Key Principle
1	**External Content Production**
a	Resources encapsulated as content packages, directly reusable within independent AHSs, delivered by a slicer service
b	AHS/slicer relationship of type master/slave
2	**Low Cost Automated Content Production**
3	**Open Corpus Content Reuse Improvement**
4	**Open Corpus Right-Fitting**
a	Correct selection and right-fitting of native open corpus resources to support recomposition within AHSs
b	Flexible slicer architecture
5	**On Demand Niche Right-Fitting**
a	AHS niche content requirements provided to slicer at run-time only
b	Selection and right-fitting of native resources based upon open domain models
6	**Content/AHS Agnostic Open-Corpus Reuse**
a	Content/AHS agnostic approach to open corpus reuse
b	Support for *search & delivery* of content packages over multiple channels
7	**Content Repurposing**
8	**Web of Slices**
a	Content packages generation process and underlying slicing resources open and shareable between independent institutions
b	Low cost slicer implementation

Table 3.1: State of the Art Influences

3.3 Slicepedia: Converting the Web into Slices

3.3.1 Introduction

The previous section summarised the opportunities and limitations discovered within the state of the art review of this thesis, along with consequential key principles derived from this analysis. This section aims to outline and refine these requirements, in order to directly support the second objective of this research aimed at the design of a content/AHS agnostic slicing service, to be named Slicepedia. A summary of high level design requirements followed by more technical requirements originating from the key principles presented earlier are hence presented, along with high level and technical architectures respectively.

3.3.2 High-Level Design Requirements Synthesis

Slicepedia aims to harness open-corpus content, available on the WWW in large volumes, and slice it in order to improve its reuse within AHSs. More specifically, in order to achieve this objective, Slicepedia should first i) accept a set of niche content requirement specifications from individual AHSs. It should ii) identify large volumes of resources, available on the WWW, of potential reuse relevancy based upon these requirements. Resources which are deemed relevant should be subsequently iii) harvested, iv) right-fitted and v) delivered to individual AHSs, as content packages immediately ready for reuse. Throughout this document, the process of performing this series of activities is referred to as *content slicing*. The resulting content packages delivered are called *slices* and clients reusing these slices are referred to as *slice consumers*.

In order to concentrate the research upon the slicing approach to open-corpus reuse, this research considers any HTML page available on the WWW as valid open-corpus resources[8].

[8]Although, HTML content clearly represents only a subset of resources available on the WWW, this format is by far the most widely used on the web. Hence it still offers a very large volume of resources for slicing and reuse within AHSs. Slicing of other file/media types such as images,

The aim is to provide a scalable solution to the content delivery needs of AHSs. The solution should be scalable with respect to three overall dimensions, namely it should guarantee the provision of i) large volumes of content, ii) at low costs, iii) suitable for multiple AHSs performing arbitrary activities. The design requirements presented below therefore fall within these three dimensions based upon what aspects of scalability they support.

Slicepedia should provide a ***service improving the reuse of open corpus resources within AHSs*** (DR 1). The slicer should be ***fully-automated and content/AHS agnostic*** (DR 2). It should be content/AHS agnostic in the sense that content should be produced i) for an non-established set of AHSs, built independently of the slicer, using techniques which do not require any predefined knowledge of either ii) the resources targeted or iii) the reuse purposes (i.e. expressed as niche content requirements). The aim of such techniques is to embrace the full diversity and volume of open corpus resources available to the AHS community by managing the unpredictability and variability of these resources and AHSs content requirements.

The ***recomposition*** of slices within various independent AHSs should be supported through the correct selection and right-fitting of open corpus resources, as well as their reuse via multiple search and delivery channels (DR 4). The provision of niche content requirements by AHSs, as well as the process of slicing itself, should be performed ***on-demand*** (DR 3); with the control over the reuse of open corpus resources, provided as a priority to slice consumers (DR 6).

So as to support larger and more diverse range of broad content requirements, the slice generation process should also be ***open and shareable*** to third party institutions (DR 5). The process of slicing should be achieved at ***low cost***, as well as the implementation of the slicer itself, so as to encourage third party institutions to collaborate (DR 7). Finally, the slicer should be designed using a ***flexible architecture*** in order to accommodate future improvements in the state of the art of relevant dependent fields of research (DR 8).

---

videos [Ketterl2008] or PDF files are reserved for future work (section section 6.3)

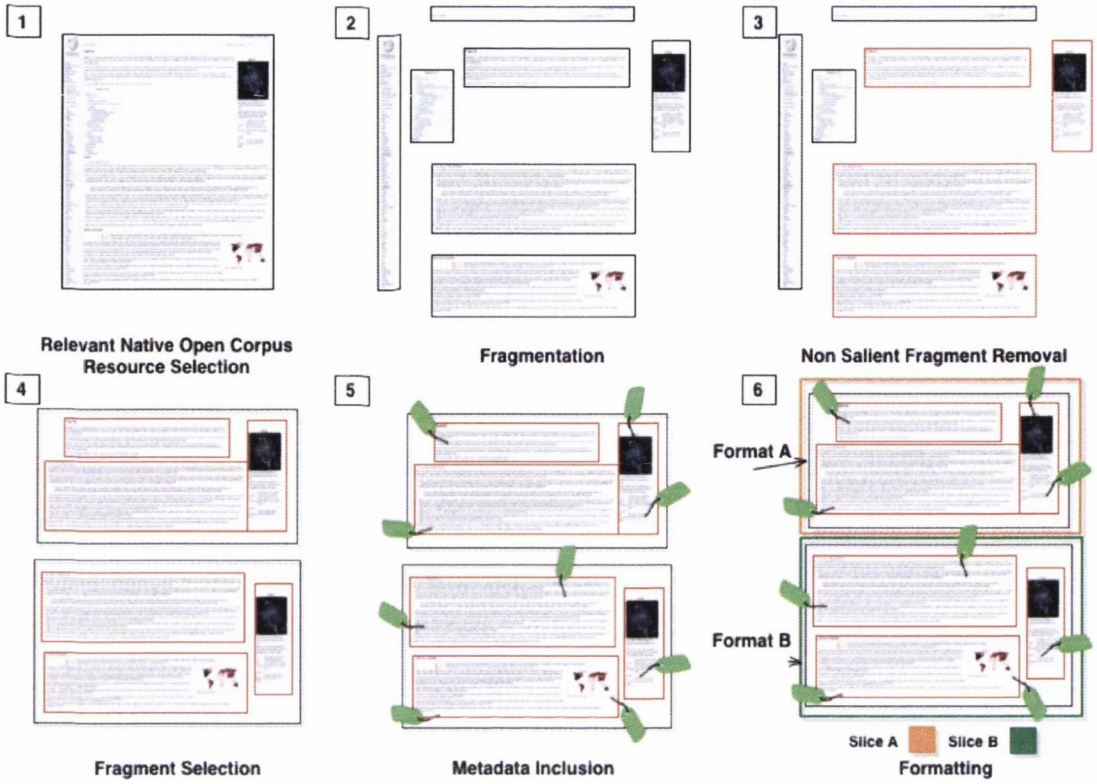


Figure 3.1: Slicepedia service viewed from individual AHSs

The resulting service described above should, from the perspective of each individual AHS, identify open corpus resources available on the web, fragment these native resources and identify fragments of relevance for each individual AHS. It should package these with relevant annotations requested by each AHS and deliver them in a format of choice (figure 3.1). The refined Design Requirements of Slicepedia, along with the Key Principles (KPs) from which they were derived, are summarised in table 3.2 and a graphical illustration of these dependencies is also available in figure A.1 (appendix A)

ID	High Level Design Requirement	KP
1	Service delivering scalable reuse improvement of open-corpus resources within AHSs	3 - 1.a
2	Fully-automated content/AHS agnostic approach to content slicing	6.a - 2
3	On-demand:	5.a
a	provision of niche content requirements	
b	slicing of open corpus resources harvested	
4	Support for recomposition of slices within various independent AHSs	1.a
a	the correct selection & right-fitting of open-corpus resources	4.a - 5.b - 7
b	the reuse of slices via multiple search & delivery channels	6.b
5	Slice generation process open and shareable to third party institutions	8.a
6	Provide a slice consumer and slicer relationship of type master/slave	1.b
7	Content slicing achieved and implementable at low cost	2 - 8.b
8	Flexible Slicer architecture	4.b

Table 3.2: High Level Design Requirements

Since the challenge of identifying web resources relevant to a particular query is a very well documented and established field of research of its own (section 2.3.5), this research considers out-of-scope the issue related to evaluating the correct identification of *relevant* open-corpus resources from the WWW and instead focuses on the harvesting, customisation, delivery and overall content slicing approach proposed by this research. As there exist many well tested alternative methods to identify open-corpus content, this research assumes a preferred method of doing so is selected by the designer of the slicing system. In order to cope with the diversity of IR approaches available, Slicepedia should hence be designed in a way which enables

the introduction of any chosen IR approach. Design Requirement 8 requiring the slicer to have a flexible architecture aims at pursuing exactly that objective.

Nevertheless, as a module identifying relevant open-corpus resources to be sliced is an indispensable part of any slicer, this component is considered an integral part of content slicing systems. For these reasons, this research assumes the open-corpus resources identified are relevant to the needs of AHSs. In this context relevancy therefore depends upon identification method selected by the designer of the slicer. What constitutes relevant content hence could go from a resource simply containing a list of arbitrary keywords in the case of traditional IR resource identification, up to a resource meeting strict topical threshold criteria in the case of more sophisticated focus crawling approaches (section 2.3.5.2).

DR 3 requires the slicing to be performed on-demand. More precisely, the on-demand nature of the slicer refers in particular to the dynamics involved between the slice consumer and slicer rather than the entire content slicing process itself. While the entire content slicing process should ideally be capable of being performed on-demand if required (see section 3.3.3.1 and 4.2), for the purpose of performance and scaling optimisations, common sense would encourage the execution of as many activities as possible a-priori of any slice request. As will be explained in the following section, the requirement for cross-institutional collaboration (DR 5) requires a compromise between a fully and partially on-demand content slicing process. The on-demand characteristic of the slicer designed for the purpose of this research hence refers specifically to the on demand provision of niche content requirements (to the slicer) and the on demand production and delivery of slices matching these requirements[9].

Due to the many complications involved when dealing with third party institution collaboration, such as linked data latency response time [OKeeffe2012] among many others (see section 6), the participation of independent institutions as part of the web of slices is also considered out of scope of this research. Nevertheless, as mentioned in

[9]Hence, whether an a-priori optimisation decision regarding the slicing process actually occurred or not is irrelevant in this context. Regardless of optimisation decisions, a slicer would still be considered on-demand, for the purpose of this research, as long as both the content request and final slices delivered did not exist at design time

DR 5, the prototype designed for the purpose of this research, should be performed with the intention of supporting such collaboration in the future.

3.3.2.1 High Level Slicepedia Architecture

Based upon the high-level design requirements outlined in the previous section, a system architecture (figure 3.3) for the slicer prototype elaborated for the purpose of this research is presented in this section.

Content published all over the globe could potentially hold greater value than what is currently derived from the single purpose intentions of original authors. More specifically, if packaged adequately, such resources could be reused within many AHSs. Slicepedia facilitates the automated reuse and delivery of resources published on the WWW. Any resource identified as relevant is harvested, fragmented, customised and delivered to individual slice consumers according to specific content requirements. Slice consumers[10], also referred to as reuse vehicles, consist of any application which provides specific content requirements to a slicer and arbitrary reuses the slices received. A use-case scenario, presented in figure 3.2, could hence be as follows:

Use-case: Suppose a thirteen year old girl, named Alice, wishes to improve her knowledge of a selected period of history. She accesses an eLearning AHS (2), which delivers resources relevant to a chosen subject and personalised to her own needs. The range of historical topics available to choose from is open, meaning that the AHS designers do not know in advance what topics will be selected by Alice or how many resources will be needed. Resources presented to Alice are personalised based upon her reading-level, time available and prior knowledge. Once the AHS has received the topic τ of interest to Alice, it makes a request for slices to Slicepedia (1).

Content requirements defined within the slice request would look as follows: the AHS would need resources i) about topic τ, ii) with a granularity for each slice sized between 2 and 5 paragraphs and focused on topic τ. The slices iii) should favour textual content over tables or bullet point lists, and iv) the reading level difficulty

[10]Within this research, AHSs are the only slice consumers considered hence both words are used interchangeably throughout the document

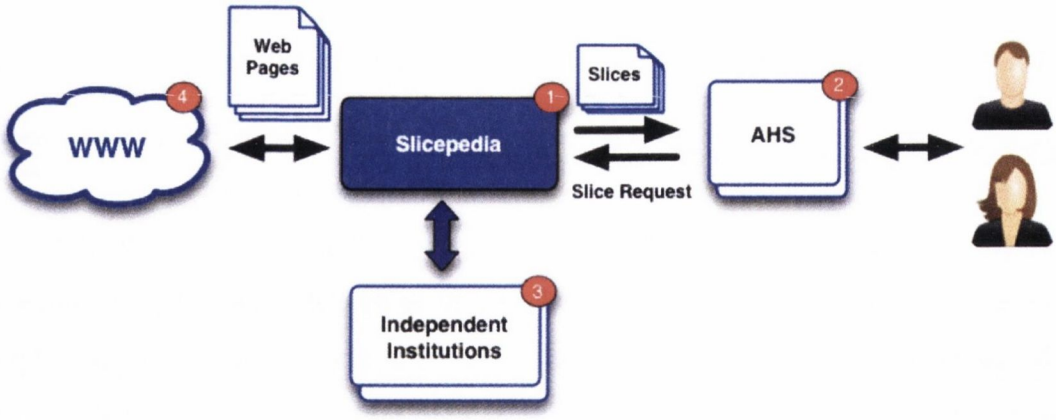


Figure 3.2: High Level Architecture (Draft)

of content within these slices should be close to 8.2 on a Flesh-Kincaid scale[11] [Kincaid1975]. Finally, v) any important historical person should also be annotated within the text and hyperlinked to a relevant Wikipedia biography. Additionally, the AHS indicates to Slicepedia that semantic analysis performed during the slicing process should only use data produced by institutions pertaining to a specific list (3).

Slicepedia identifies relevant resources from the WWW (4), which it subsequently slices to match the requested content requirements as best as possible, and delivers back to the AHS. The slices are then recomposed by the AHS. Thanks to annotations provided by Slicepedia, the AHS also displays hyperlinks within the text of any personality it is aware Alice hasn't come across already and hides all the others.

Based upon the previous example, a slice can hence be defined as: *customised content consisting of fragment(s) (originating from pre-existing document(s)) assembled, combined with appropriate meta-data and right-fitted to the specific content requirements of a slice consumer, with various application-specific content-reuse intentions, different (or not) from the original author intentions.* Slices can contain other slices and can be reused or re-composed with many slices.

[11] Reading difficulty of 8.2 on a Flesh-Kincaid reading score scale indicates the text is expected to be understandable by an average student in 8th grade in the USA (between 12-14 years old)

3.3.3 Technical Requirements

The previous section presented a series of high-level design requirements from influences presented in the state-of-the-art chapter. This section will now presents technical requirements for Slicepedia aimed at providing more technical precision to individual high-level design requirements. A summary of these technical requirements is outlined at the end of this section.

As mentioned in Design Requirement (DR) 7, the slicer implementation should be performed at low cost. It is an approach of this research hence to prioritise the use of ***free open-source or Software as a Service (SaaS)*** (TDR 1) technologies when possible in order to lower implementation costs and therefore encourage various institutions to build their own version of such a slicer and participate in collaboration efforts. If any enterprise software is selected, open-source alternatives available to the community should be identified.

According to the previous section, Slicepedia is required to provide a fully-automated (DR 2), scalable (DR 1) and content/AHS agnostic (DR 2) approach to content slicing for open-corpus resources. Since an important step of the slicing process consists in fragmenting resources, from the point of view of fragmentation, these requirements translate into the ability to process i) large volumes of diverse web resources[12], ii) without any human intervention or iii) prior knowledge of resources being targeted for fragmentation and iv) within a reasonable amount of time. For the purpose of this research, a slicer capable of processing hundreds of thousands of pages within a matter of hours[13], is considered to be of adequate performance. Hence, based on this set of high-level requirements, and according to the range of algorithms identified within the state-of-the-art chapter, ***content fragmenter characteristics*** of importance when selecting an appropriate algorithm for Slicepedia should consist of the following; The fragmenter should process i) large volumes of pages at high speed, fully-automatically (TDR 3.a). It should ideally be ii) domain and language

[12]diverse resources available in various languages, covering various subject domains

[13]Using a standard mainstream computer for example CPU: 2.8Ghz Intel Core 2 Duo, Mem: 4GB 1067 MHz DDR3

independent (TDR 3.b), meaning it shouldn't require any kind of semantic content analysis in order to produce fragments with high accuracies and should be iii) content type[14] independent (TDR 3.c). Finally, since a large volume of unknown pages should be targeted by the slicer, each page could theoretically be harvested from different websites. For this reason, the fragmenter to be used as part of Slicepedia should consist of a iv) page-level fragmenter as opposed to site-level (TDR 3.d). A site-level fragmenter would require to harvest multiple sets of additional pages for each open-corpus resource targeted by the slicer, which would exponentially increase the already large number of pages to process. Since the set of pages prior to slicing is unknown and hence the number of possible DOM layout patterns is infinite (section 2.4.3), fragmentation should also occur without the need for interpreting the v) meaning (TDR 3.e) or vi) structure of tags within a page (TDR 3.f). Finally, since Slicepedia should produce slices at various levels of granularity (DR 4.a), the ability to vii) predictability control the sizing of resulting fragments is also important (3.g).

DR 4.a requires Slicepedia to provide right-fitting control of open-corpus resources to AHSs in order to support their recomposition. The ***right-fitting dimensions*** considered should hence be performed with respect to *i*) content granularity (TDR 5.a) and *ii*) style (reading difficulty, table, bullet points) (TDR 5.b). In order to support page adaptation by AHSs, it should also provide *iii*) the insertion of chosen annotations within the content supplied (based upon request) (TDR 5.c). Within this research, granularity refers to the combination of both the sizing[15] of content and its focus. This research defines content focus as the relationship between an arbitrary focal point within a piece of content, and the sizing of the area to which it belongs. A content package correctly focused on a topic, for example, would only contain content about the topic in question. Content considered unrelated to that topic would be removed. In other words, the sizing of the content should match the area covered by this focal point. Granularity control, including content focus,

[14]Content type in this research refers to encyclopaedia articles, news articles, product and forum pages etc.)

[15]Content sizing simply refers to the size of a particular content package. This size might be expressed by the number of words or paragraphs contained within the text.

enables AHSs to target pertinent parts of a native resource to reuse as opposed to the entire resource. Slicepedia should therefore provide the ability to focus resources based upon topical elements of a page as well as annotations introduced. Topical content focus should be performed based upon either specific keywords encountered within a resource, as well as broad domain coverage within that resource.

In order to promote openness and collaboration, both the generation of slices as well as the underlying resources needed to produce slices must be accessible between institutions (DR 5). For this reason, Slicepedia should make ***extensive use of RDF*** as an underlying data representation mechanism (TDR 7). As mentioned in section 2.4.2.3, this metadata standard is now widely used as a method to share the description of concepts or model information on the internet. The representation in RDF of fragments and meta-data, which form the fundamental building blocks of slices, would enable various institutions to publish meta-data and annotations, produced using their own algorithm, as linked data and hence enable the seamless combination of data from multiple sources at no extra cost. Finally, according to design requirement DR 4, slices must be directly reusable as part of various independent domain models. Using RDF would enable slices to be seamlessly connected to an abundance of linked data nodes, which could thereafter be used by AHSs [Steichen2011a] as ***linked data domain anchor points*** (connected using same-as relationships to internal domain ontologies). RDF would additionally support the use of inference over concepts and meta-data related to slices by using the expressivity of such standardised ontologies (section 2.4.2.3). The result of opening and sharing the building blocks necessary for the production of slices could eventually lead to a so-called web of slices[16].

The support for multiple slice search and delivery alternatives was identified as a requirement in the previous section (DR 4). As presented in section 2.3.4, various AHSs have already started leveraging the wealth of resources available on the WWW

[16]A *web of slices* would consist of a network representing relationships between meta-data and fragments, leading to a range of potential slice content packages. A visual representation of a hypothetical web of slice is represented in figure 4.11 (section 4.4)and a full definition is available in the glossary

through the use of IR technology. Since keyword-based queries represent the de facto search mechanism underlying IR systems, Slicepedia should at a minimum support this ***slice selection mode*** (TDR 4.a). Moreover, as slices will be incorporated within independent domain models through the use of domain anchors, the ability to identify slices based upon their relation to specific domain concepts through the use of conceptual selection should be made available to AHSs (TDR 4.b). Both keyword and conceptual selection should also be combinable (TDR 4.c). Since Slicepedia is required to use RDF as its data representation mechanism, the slicing data[17] ***search and delivery*** as well as meta-data inference should additionally support the use of the *SPARQL* queries (TDR 6.a). Finally, a growing number of services and web applications are now moving to the cloud. Cloud hosting services[18] provide resizable computing capacity which makes web-scale computing easier for developers to manage. The vast majority of web-services nowadays are designed using the REpresentational State Transfer (REST) model [Fielding2002], which enables the loose coupling of transactions between web services. As AHSs naturally tend towards serving more diverse needs of larger number of users, one can predict the move of mainstream AHSs towards these cloud/RESTful computing platforms. Hence Slicepedia should additionally provide a RESTful web-service interface (TDR 6.b) and support slice delivery integration with one or more cloud platforms (TDR 6.c).

This research focuses solely upon open-corpus resources available in English, as a first step and reserves multilingual slicing for future work (section 6.3). Nevertheless, since open-corpus resources are by nature multilingual, assuming the reuse of content through slicing is validated by this research, multilingual slicing will represent an important requirement of future developments. For this reason, Slicepedia should be designed in such a way as to easily permit future improvements which support multilingual slicing. Components used for the development of this prototype should therefore be selected based upon their ***multilingual support*** (TDR 2). In the event a component selected does not support multilingual functionalities, alternatives

[17]What constitute slicing data is formalised in section 4.2.3

[18]Google app engine https://developers.google.com/, appengine/, Amazon Cloud Computing http://aws.amazon.com/ [Accessed: October 3, 2014]

to these components supporting other languages should be provided.

An important part of slicing involves the analysis of open-corpus resources, Slicepedia should make full use of existing NLP algorithms (section 2.4.4) and combine them within a NLP pipeline. Additionally, since Slicepedia should be content/AHS agnostic (DR 2) and support subsequent improvements towards multilingual slicing, this means the language of the open-corpus resources targeted for reuse will be unknown at design time. For this reason, the approach adopted by this research should use ***real-time adjustable NLP pipelines*** (TDR 8) based upon the language of native resources processed by the slicer. Moreover, AHSs require the support of varying sets of annotations based upon the task they are serving (section 2.3.2), the NLP pipeline should enable designers to easily ***plug in/out various NLP algorithms*** based upon the set of AHSs considered for slice consumers.

A summary of Slicepedia's technical requirements is presented in table 3.3, along with the corresponding high level design requirements they support. A graphical representation of these dependencies is also available in figure A.2 (appendix A). No technical design requirements were derived from high level design requirements DR 3, DR 6 and DR 8, as these were directly supported through implementation decisions (section 5.3).

ID	Technical Design Requirement	DR
1	Free Open Source or SaaS Usage	7
2	Support for Multilingual Slicing Improvement	2
3	Fragmenter characteristics:	
a	high speed and fully-automated processing of large volumes of pages	2
b	domain and language independent	2
c	content type agnostic	2
d	page-level fragmenter	1
e	tag meaning independent	2
f	tag structure independent	1 - 2
g	predictable fragment sizing control	4.a
4	Slice Selection Modes	4.a
a	keyword-based selection	
b	conceptual selection	
c	combined keyword/conceptual selection	
5	Right-fitting of open corpus material with respect to:	4.a
a	Granularity (Sizing & Content Focus)	
b	Style	
c	Support for AHS page adaptation through annotation insertions	
6	Slice Search & Delivery Channels	4.b
a	SPARQL	
b	REST API	
c	Cloud platform	
7	RDF Storage and Linked Data domain Anchor Points	5 - 4.a
8	Real-time Adjustable & plug in/out NLP PipelineAlgorithms	2 - 8

Table 3.3: Technical Design Requirements

3.3.3.1 Technical Architecture

Based upon the requirements defined in the previous two sections and the objectives of this thesis a technical system architecture was defined for Slicepedia. The architecture presents the overall procedure involved with performing open or closed content slicing in general. Chapter 4 will describe how the slicer architecture, presented within this section, can be implemented to specifically pursue open-corpus content/AHS agnostic slicing.

As depicted in figure 3.3, Slicepedia is designed as a novel pipeline tool chain combining various individual components consisting of a: 1) *content harvester*, 2) *structural fragmenter*, 3) *semantic analyser* and 4) *slice generation unit.*

Designing Slicepedia using a pipeline architecture enables each component, presented above, to be replaced with better versions as the state-of-art in each relevant field progresses. This flexible architecture (DR 8) allows for simple interfaces between each component to be defined, which minimises any changes needed in other components of the system.

The purpose of the *content harvester* (1) is to *identify* and *harvest* relevant resources

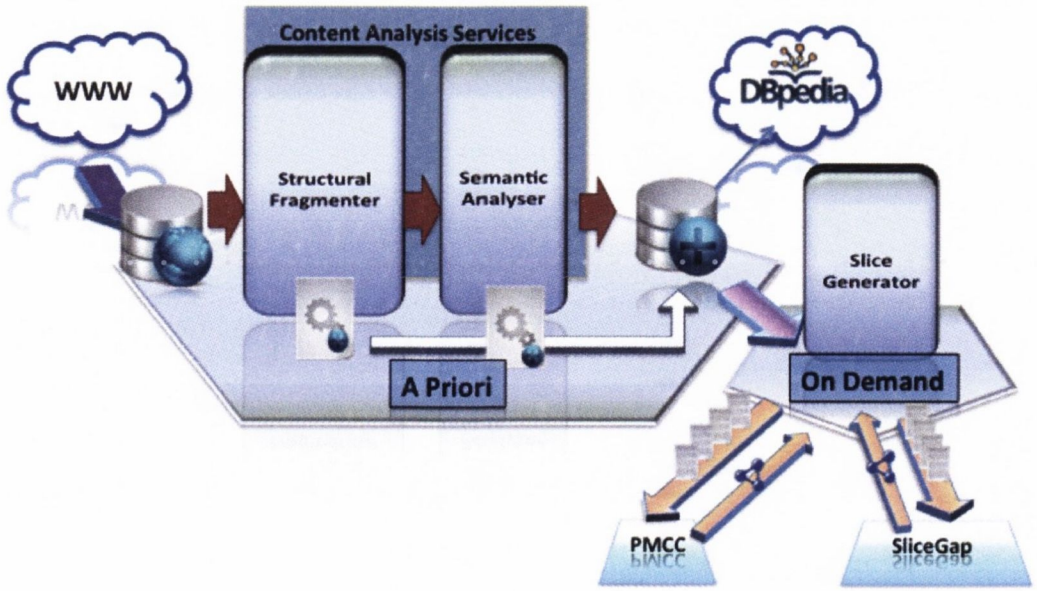


Figure 3.3: Slice Architecture Pipeline

from the web, in their native form.

Relevant resources are then processed through the *structural fragmenter* (2), which analyses and fragments native resources into structurally coherent pieces (such as menus, advertisements, main article) as described in step two of figure 3.1. Structural meta-data, such as the location of each fragment within the original resource, is extracted and stored in a meta-data repository (3). This phase is critical since, as mentioned previously (section 2.2.3), maximising the reuse potential of a resource involves the ability to reuse selected parts of documents, which in turn, depends upon correctly identifying individual sections of pages to produce SCI fragments. Any resulting structural inconsistencies[19] will have a direct impact upon the overall quality of reuse, regardless of the performance of subsequent steps.

Once a set of foundation fragments is produced, each fragment is processed by a *semantic analyser* (4) which produces discerning meta-data, supporting its identification and selection for reuse by individual slice consumers. While meta-data produced previously only focused on structural aspects of resources harvested, meta-data produced by this component instead focuses upon the semantic[20] attributes of the resources.

What constitutes discerning meta-data is highly dependant upon the reuse intentions of each slice consumer (section 2.3.4.2). Hence an inappropriate production of meta-data could therefore lead to the potential for very low reuse across consumers. Great care must therefore be taken during the selection of suitable annotators in order to support the broadest needs possible of targeted consumer use cases.

Since DR 2 requires Slicepedia to be content/AHS agnostic, the range of targeted documents is hence opened. As a consequence, the resulting open content model available to slice consumers should not be restrained by any predefined AHS subjective domain model. For this reason, and as mentioned in the previous section, Slicepedia disentangles domain modelling from its open content model and instead provides

[19]such as parts of original menus erroneously broken in two and merged with paragraphs or various titles merged together

[20]The word *semantic* here is used in the broadest sense of the term. Meta-data referred to as semantic meta-data might include topic covered, writing style, or the difficulty level of content

anchors to Linked Data concepts as a foundation for any domain model chosen by individual slice consumers. As required by TDR 7, fragments and annotations produced by Slicepedia are also available as Linked Data.

Once individual fragments and metadata annotations are available and whenever slice requests are received, a *slice generation unit* (5) identifies the most pertinent combinations of foundation fragments and meta-data possible, and produces a set of right-fitted slices matching as best as possible content requirements expressed within the slice request. Slices produced are then ordered based on how well they fit requested content requirements. An arbitrary number of slices are finally sent back to the slice consumer to be reused. The resulting overall array of possible adjustments (such as the extent of control over granularity, style and annotation) a slicer can offer upon an open-corpus resource is referred to as a Content Adaptation Spectrum (CAS).

Each individual component of the pipeline, from the harvesting of open-corpus resources to the delivery of customised slices, can be performed on-demand. However, in order to support performance and scaling optimisations (as mentioned in section 3.3.2), when possible the first three components of the pipeline can also be executed a-priori of any slice consumer request. Whenever broad content requirements (as opposed to niche content requirements) of potential slice consumers can be predicted in advance of any AHS interaction, the *content harvester* can pre-emptively identify potentially relevant resources on the WWW and forward them to the *structural fragmenter* and *semantic analyser* modules. Whenever, slice consumer requests are finally received by the slicer, the *slice generation unit* can either produce slices on-demand based on these pre-emptively processed resources or alternatively request additional resources to be harvested and processed. Within this context, the slicing is still considered on-demand, since niche content requirements (as opposed to broad content requirements) needs of slice consumers are unknown at design time and slices delivered do not exist prior to slice request being submitted. As will be mentioned in section 4.4, the requirement for Slicepedia to support third-party collaboration (DR 5), is an example of when such an a-priori optimisation decisions must be made.

3.3.4 Summary

This section presented the overall design requirements of Slicepedia, a service which provides the automated harvesting, customisation and delivery of content packages from open-corpus resources to AHSs. The following section, presents the design of a more detailed process, referred to extensively within the rest of this document, which involves the structural analysis of open corpus resources.

3.4 Densitometric Content Fragmentation

3.4.1 Introduction

The Densitometric Content Fragmentation (DCF) algorithm, developed by [Kohlschutter2008a] and initially introduced in section 2.4.3.5, was selected[21] from all the content fragmentation approaches reviewed, as the most promising algorithm susceptible of meeting the fragmentation design requirements (TDR 3) enunciated in section 3.3.3. This algorithm was eventually implemented and evaluated with respect to these requirements (presented in section 5.2). For these reasons, a detailed description of this algorithm is provided in this section. Following this evaluation, modifications to this algorithm were performed in order to overcome some shortcomings identified during this analysis. This section hence additionally serves as a basis for describing the modifications performed upon the DCF algorithm variation presented in the following section. This section therefore presents the fundamental concepts related to DCF in general, required for the purpose of understanding the subsequent analysis and algorithm design decisions.

3.4.2 Traditional Densitometric Content Fragmentation

Within the context of densitometric fragmentations, leaf nodes (within the DOM tag tree representation of a page) of a resource are converted into a one dimensional array of text density values (figure 3.4). The ***text density*** ρ_{τ_x} (equation 3.1) of a tag τ_x is defined as the ratio between the number of tokens and the number of lines within τ_x.

$$\rho_{\tau_x} = \frac{\sum Tokens_{\tau_x}}{\sum Lines_{\tau_x}} \tag{3.1}$$

A line is defined as a group of successive characters, with a total character number equal to an arbitrary ***word wrapping*** value W_x. Kohlshutter suggests word wrapping value to be fixed at a value of 80. Tags containing only one line of

[21] Details as to why this algorithm in particular was selected are provided in section 4.3.3.4.1

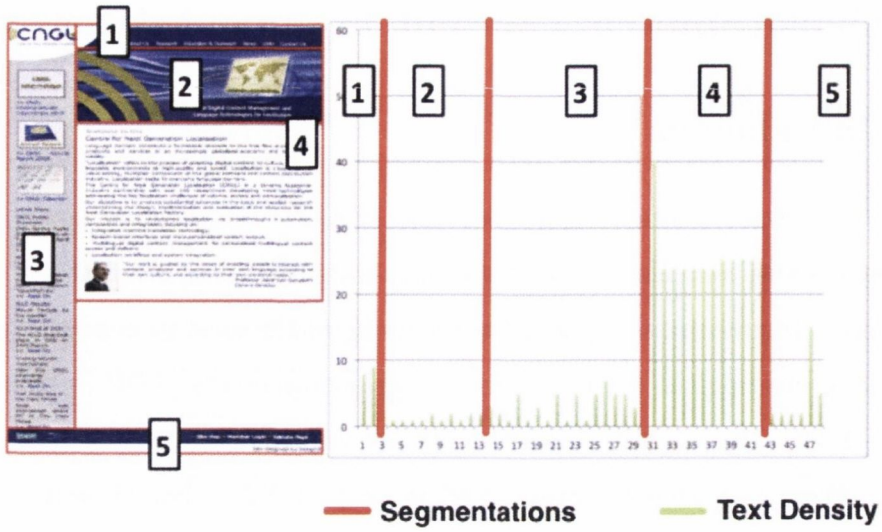


Figure 3.4: Densitometric Page Representation

text are assigned a text density value equal to the number of tokens it possesses. Although a line might appear as an arbitrary notion, figure 3.4 shows how sharp changes in text density correspond relatively well to desired fragmentations. DCF therefore consists in the process of identifying such variations in text density and correlating them with fragment boundaries.

DCF algorithms proceed in detecting these variations by considering each leaf tag text density value as one individual atomic block (or fragment). Each page is hence converted into a single block array. DCF algorithms subsequently iterate through this array by selecting a set of individual blocks and then combining them together into larger compounded blocks to form a new block array (composed of atomic and compounded blocks). The number of consecutive individual blocks compared before fusion is referred to as the ***block window size***. The process of combining adjacent blocks together is referred to as ***block fusion***. This process is achieved by joining the lines of two blocks α and β inside a new block γ, such that γ spans from the first line of α to the last line of β. α and β are then removed and replaced by the new block γ. Iterations through the block array are performed multiple times, by fusing compounded blocks together, until final fragments are created.

At the time of writing, existing DCF algorithms differed mostly upon the criteria

selected for fusion, as well as how adjacent blocks are selected for comparison prior to fusion. All DCF algorithms however finish whenever an iteration is performed within the entire block array, without the occurrence of any fusion. The four DCF variations originally presented by [Kohlschutter2008a] are described below:

Plain fusion

Plain fusion for instance, only considers pairs of adjacent blocks at a time (equivalent to a block window size equal to two). If the text density difference Δ_ρ (equation 3.2) of the pair is smaller than an arbitrary threshold V_{max}, the blocks are fused and the resulting compounded block produced is compared to the next one and so on, as described in figure 3.5.

$$\Delta_\rho(i, i+1) = \frac{|\rho(i) - \rho(i+1)|}{max(\rho(i), \rho(i+1))} \tag{3.2}$$

Smooth fusion

Smooth fusion extends the previous algorithm by considering instead the text density of the preceding and succeeding block (block window size equal to three). If the text densities of preceding and succeeding blocks are identical or higher than a blocks text density, all three blocks are fused together into a compounded block.

Rule-based fusion

Rule-based fusions on the other hand, attempt to augment the previous algorithms by taking into account the meaning of specific tags (titles, tables etc.) in order to infer structural composition of pages. They augment a DCF algorithm with a predefined set of rules similarly to wrapper-based content fragmentation algorithms discussed within the state-of-the-art review (see section 2.4.3.3). Whenever such an arbitrary tag is encountered, a *block fusion* or *block gap* is performed regardless of text density values. However, since taking into account the meaning of tags violates TDR 3.e, the use of rule-based fusion should be minimised (ideally never used) within the context of a slicer pipeline.

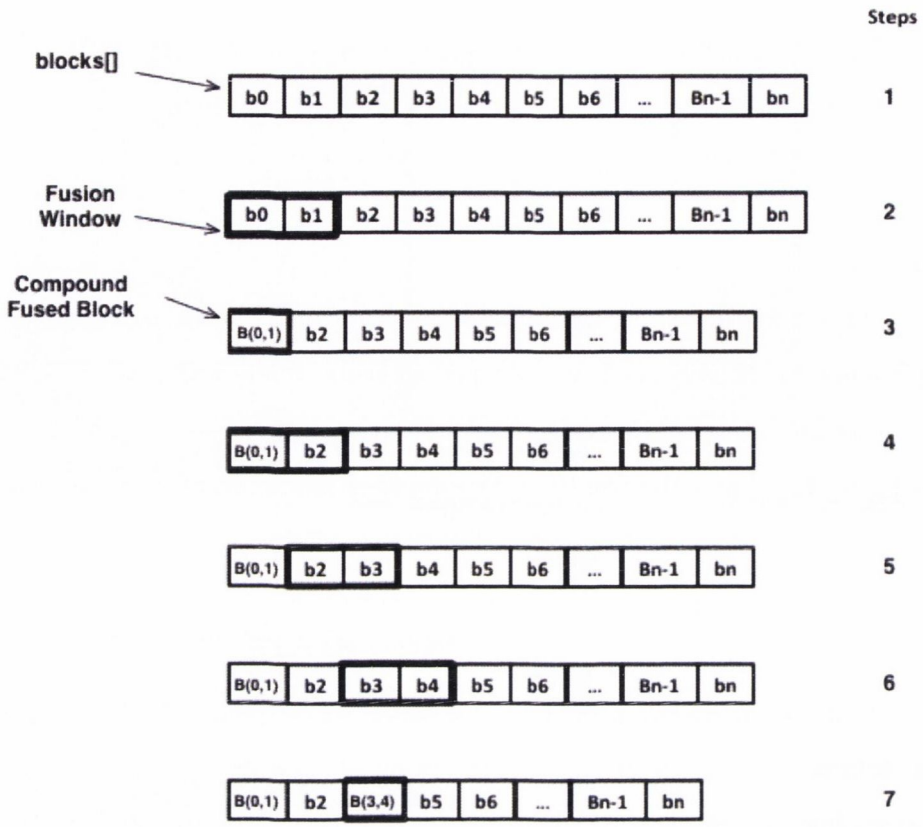


Figure 3.5: Plain Fusion Representation

Smooth rule-based fusion

Finally and as its name suggests, smooth rule-based fusion consists of the two previous fusion variations, merged together. Hence this type of fusion uses a set of tags enforcing block fusion and block gaps, and considers text densities of preceding and following blocks when no such tag is encountered.

3.4.3 Greedy Densitometric Content Fragmentation

As explained in the previous section, the DCF approach to content fragmentation was selected as the most promising fragmentation technique for the purpose of content slicing. The analysis performed in section 5.2 however highlighted some drawbacks with respect to this approach. This section therefore presents an alternative version of the original DCF algorithm presented previously and originally developed by [Kohlschutter2008a]. This variation uses, as a foundation, the same fundamental

DCF principles described in the previous section proposed by Kohlschutter and introduces some modifications during the block array iterations phase of the original algorithm. The modifications aim to address in particular i) the time performance variations (with respect to fragment size output) discovered in the analysis and also ii) take advantage of local text density attributes of different regions of pages to further refine the accuracy of the algorithm.

This greedy algorithm, developed for the purpose of this research, hence proposes to stabilise the time performance of plain fusion DCF across sizing outputs, by reducing the number of iterations through the densitometric block array. Since the original DCF algorithm proceeds via a bottom up approach (successively merging individual atomic blocks into larger compounded blocks), larger sizing output requirements will lead to a higher number of iterations within the array. The greedy variation hence proposes to replace the fixed block window size (section 3.4.2) of the original algorithm with an ***expanding window size***. A greedy behaviour drives this window expansion (see figure 3.6), starting initially with a block window size equal to two (as for plain fusion) at the start of each iteration and subsequently increasing.

During the analysis carried out upon the original DCF variations (section 5.2, multiple manual inspections of densitometric page representations within the corpus constantly revealed clear distinct densitometric regions within individual pages. This property of pages isnt taken into account by existing DCF variations. The assumption is that local adjustments of threshold values to specific regions of a page (such as menus, or user comments with low average densitometric values in comparison to an article with high values) could perhaps also improve the fragmentation accuracy of regions of pages with differing local densitometric characteristics. This algorithm hence also differs from existing approaches (which use non-variable thresholds) by using a ***variable threshold*** V_{max} ***value***, automatically adjusted with respect to local regions of a page, based on densitometric values of blocks currently selected within the window. The replacement of a fixed block window size with a variable one should theoretically make most fusions occur in early iterations and therefore stabilise time performances for high V_{max} values.

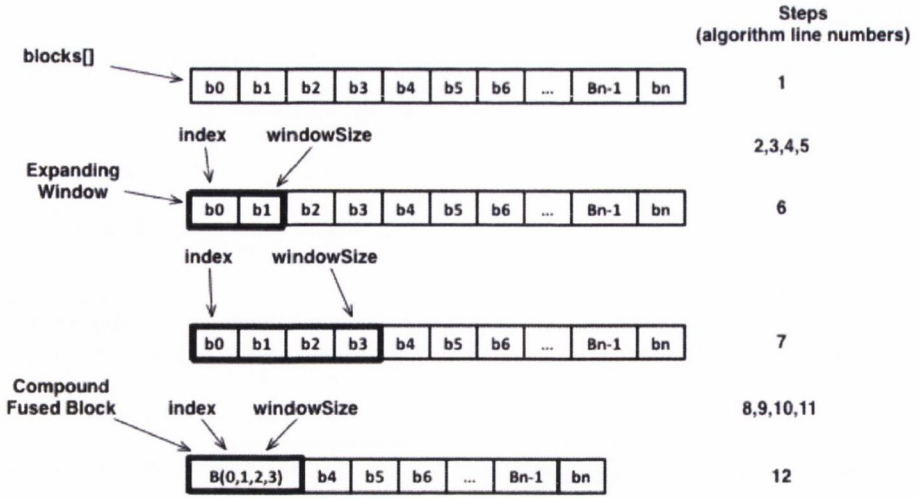


Figure 3.6: Greedy Algorithm Description

A detailed description of the algorithm is provided in algorithm 1. As mentioned above, the Greedy DCF algorithm, initially starts the fragmentation process as performed during a plain fusion fragmentation. In other words, each page processed is converted into a block array with text density values corresponding to each leaf-tag within the resource. The threshold V_{max} value is also assigned an arbitrary value and the block window size is set to two (step 1 in figure 3.6). Text density differences of blocks contained within the window are then compared to V_{max} as during the plain fusion algorithm (step 6). If the difference in text densities is below that threshold however, blocks are not immediately fused together. Instead, the block window size is incremented by one, and V_{max} is assigned the average value of block text densities contained within the window. The text density difference between the new block introduced within the window and the previous one is thereafter compared to the new V_{max} value computed (step 7). If their difference is below that value, the window size is incremented and the same procedure follows. Whenever this difference is above the average V_{max} value, the new block is ignored from the set contained within the window and all other blocks are fused together into one (step 12). When this event occurs, the window size and V_{max} values are assigned their original values and the algorithm compares the next two blocks with this new window of size equal to two (step 6)

Algorithm 1 Greedy Densitometric Algorithm

```
1: procedure GreedyFusion(blocks[], defaultV_max)
2:     fusedBlocks ← true;
3:     while fusedBlocks do                    ▷ New Array Iteration Initialisation
4:         fusedBlocks ← false;
5:         index ← 0;
6:         while index < blocks[].size do                    ▷ Array Iteration
7:             windowSize ← FusableWindowSize(index, blocks[])
8:             if windowSize > 0 then
9:                 blocks[] ← Fuse(index, windowSize, blocks[])
10:                fusedBlocks ← true;
11:            end if
12:            index + +;
13:        end while
14:    end while
15:    return blocks[];
16: end procedure
```

```
17: function Fuse(index, windowSize, blocks[])
18:     tempBlocks[] ← blocks[index, index + windowSize];
19:     ρ_blocks[index] = ∑Tokens_tempBlocks[] / ∑Lines_tempBlocks[];
20:     blocks[index] ← ρ_blocks[index];
21:     remove blocks[index + 1, index + windowSize];
22:     return blocks[];
23: end function
```

```
24: function FusableWindowSize(index, blocks[])
25:     V_max List.add(default V_max);
26:     windowExpanded ← true;
27:     blocksToMerge ← 0;
28:     greadyIndex ← index + blocksToMerge;
29:     while windowExpanded && (greedyIndex + 1 ≤ blocks[].size) do
30:         windowExpanded ← false;
31:         densitoDif ← Δ_ρ(blocks[greedyIndex], blocks[greedyIndex + 1]);
32:         if densitoDif > avg(V_max List) then
33:             V_max List.add(densitoDif);
34:             blocksToMerge + +;
35:             windowExpanded ← true;
36:             greedyIndex ← index + blocksToMerge;
37:         end if
38:     end while
39:     return blocksToMerge;
40: end function
```

3.4.4 Densitometric Fragmentation Summary

This section presented the DCF approach to open corpus fragmentation, which was selected, among algorithms reviewed in section 2.4.3, as the most promising algorithm with respect to technical design requirements enunciated in section 3.3.3. Algorithms of existing DCF approaches, as well as a new variation designed specifically for the purpose of this research, were described in detail so as to support an understanding of their analysis in section 5.2 of this thesis. The subsequent chapter provides a detailed explanation with respect to why this approach was selected among existing alternatives.

3.5 Conclusion

This chapter presented the design decisions and requirements underlying the slicer prototype to be implemented for the purpose of this research, called Slicepedia, which can harvest, fragment and right-fit open-corpus resources for reuse within AHSs. Proposed high-level and technical architectures of Slicepedia were described along with an example use-case scenario. The following chapter will now present the implementation decisions taken in order to fulfil the design architecture and requirements presented above.

Chapter 4

Implementation

4.1 Introduction

The previous chapter of this thesis described the design of Slicepedia, a slicing system aimed at improving the reuse of open-corpus material by right-fitting each resource to content requirements of AHSs. The design was derived from influences observed in the state-of-the-art and refined in high-level and technical requirements. This chapter now describes the implementation of these design decisions, in order to produce a slicer prototype performing open-corpus harvesting, customisation and delivery to slice consumers, and thus contribute to objective two of this thesis.

This chapter builds upon the architectures presented in the previous chapter and begins by describing in section 4.2 the overall arrangement and coordination of components supporting the Slicepedia prototype implementation. Each component is then described in section 4.3 in more detail based on how they support each design requirement enunciated in the previous chapter.

It is a strategy of this research to utilise open source software or free SaaS tools (TDR 1), when possible, in an effort to avoid reinventing the wheel. Rationales underlying decisions taken to choose one specific third party components as opposed to others are hence provided. All of the modules described in this chapter were implemented for the purpose of this research. An important part of the implementation efforts

carried out for this research involved combining existing components (i.e: web crawler, SPARQL database, NLP algorithms or web servers) together to produce a slicer tool chain. To achieve this goal, several toolsets, frameworks and existing NLP algorithms were used. When existing tools were used, these are referred throughout this chapter, in each appropriate section dedicated to individual parts of the tool chain. Additional modules and NLP algorithms were also implemented specifically for the purpose of this research, in order to support the integration between components, and extract specific data from individual documents analysed.

4.2 Slicepedia Overall Architecture

4.2.1 Introduction

The architecture of Slicepedia, as illustrated in figure 4.1, facilitates the harvesting, fragmentation, customisation and delivery of open-corpus resources to AHSs.

As mentioned in section 3.3.3.1, the prototype slicer implemented for the purpose of this research achieves this by implementing various units each addressing one of these tasks. These components consist of a: 1) *resource harvester unit*, 2) *analyser unit* (including structural and semantic analysis) and 3) *slice generation unit*. The following sections present an overall description of how each of these components coordinate with each other.

4.2.2 Slicepedia Requests Coordinator

This architecture is designed as a pipeline and for this reason, components can only be run successively, in the numerical order specified above. Nevertheless, the entire pipeline can be run based upon three different Slicepedia Request (SR) types. For this reason, an additional component, named *Slicepedia Requests Coordinator (SRC)*

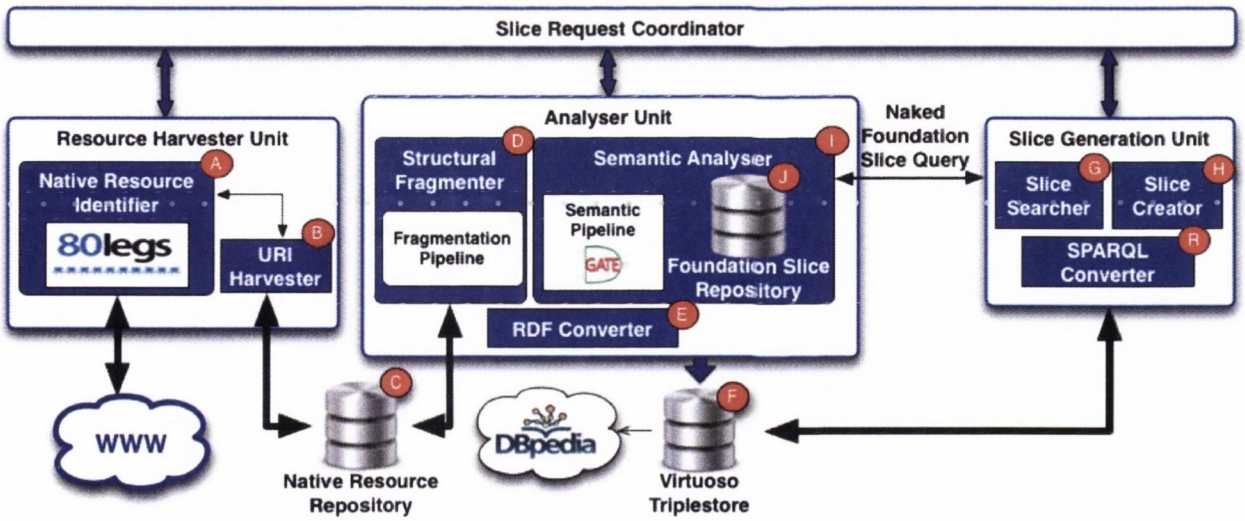


Figure 4.1: Slicepedia Implementation Architecture

module, is responsible for coordinating each component together based upon what Slicepedia request is being performed. Slicepedia can be requested to either

SR_1) *prepare open-corpus resources* for reuse or SR_2) *generate slices* (slice request). When performing a slice request, if slices are to be generated from open-corpus resources which haven't been prepared yet then Slicepedia performs SR_3) both *the preparation of resources* and *slice requests* all at once. The last option is recommended only when targeting specific pages (or small number of resources) for immediate slicing and reuse. In the majority of cases however, and as mentioned in section 3.3.3.1, for optimisation purposes and when possible, the ability to perform as many tasks a-priori of AHS interaction will be preferred. For this reason, request SR_1 will only run the first two units of the pipeline (i.e. harvesting, fragmentation and semantic analysis), later on followed by multiple slice requests. When Slicepedia is requested to prepare open-corpus resources for reuse, resources targeted for slicing are processed and ready to convert into slices before any AHS interaction with the slicer. Slices can later-on be produced on-demand based on AHS content requirements, by performing slice requests (see remark on On-Demand Slicing design requirement). As will be described in section 4.4, the first two Slicepedia requests could also be used when slicing resources in collaboration with other institutions. Within this context, the semantic analyser step also becomes optional.

Finally, the SRC module is also responsible for separating resources targeted for slicing into smaller unit batches. This feature enables an optimisation of the Slicepedia service response time. It does so by removing the need for slice consumers to wait until all open-corpus resources targeted have been processed and instead accepts slice generation requests, for a subset of open-corpus resources, while the rest of open-corpus resources are been processed (harvested and analysed).

4.2.3 Slice Request Procedure

This section presents the procedure carried out by Slicepedia when a slice consumer initiates a slice request. In order to provide a complete understanding of the underlying interaction between all components, this section will assume the Slicepedia

request performed is of type SR_3, meaning that the slicer does not yet possess the resources targeted for reuse by the slice consumer and hence the three units constituting the pipeline (section 4.2) are all run successively.

A slice request[1] is encapsulated within a ***SliceQuery object*** (section 4.3.4.2). This object represents the set of specific content requirements selected by the slice consumer, among the CAS provided by the slicer. Once Slicepedia has received such a request, the Slicepedia Requests Coordinator (SRC) forwards the request to the *resource harvester* unit. Since it is assumed that at this stage Slicepedia does not contain any resources, the slice query must contain at least either a set of keywords or list of preferred open-corpus repository urls to crawl.

The ***resource harvester unit*** subsequently passes the set of keywords or the list of open-corpus repository urls to its *native resource identifier* module (A), which identifies relevant[2] web pages. Once an arbitrary number of relevant web page urls are identified, these are forwarded to the *url harvester* module (B), which simply downloads the resources in their native form, into the *native resource repository* module (C). The native resource repository used for this implementation of Slicepedia simply consists of a directory on the local file system (section 4.3.2). Paths pointing to each native resource are then forwarded to the analyser unit.

The ***analyser unit*** starts processing resources through the *structural fragmenter* module (D). This module analyses each resource contained within the native resource repository in order to produce sets of structurally coherent fragments along with structural meta-data. Fragments and meta-data produced, are then converted to RDF statements using the *RDF converter* module (E) (implemented for the purpose of this research) and stored in a *Virtuoso*[3] *repository* (F).

Although no semantic analysis has being performed upon fragments at this stage, once an initial set of fragments and meta-data have been produced and stored, the ***slice generation unit*** can be run. The CAS made available by the slicer at this stage is very limited of course, since only structural meta-data is available. Nevertheless

[1] A detailed example of what would constitute such a slice request is presented in section 4.6.

[2] What constitutes relevant pages is discussed in section 4.3.2

[3] http://virtuoso.openlinksw.com/

very simple slices, relying solely on structural aspects of native resources processed, can already be produced. In order to generate such slices, the slice generation unit forwards the slice query object received from the SRC to the *slice searcher* module (G) which returns a set of *SliceHits*. A SliceHit[4] is simply a Plain Old Java Objects (POJOs) that represents a possible match between a set of content requirement expressed as a SliceQuery object and one unique slice.

These SliceHits which are either forwarded to slice consumers for inspection or directly transferred to the *slice creator* module (H). For each SliceHit received, the latter constructs the corresponding Slice based upon the instructions contained within the SliceHit (4.3.4.4). The resulting set of slices produced is finally sent back to the SRC, which forwards it back to the slice consumer.

In most cases however, prior to running the slice generation unit, and following the structural fragmenter module, the SRC triggers *semantic analyser* module (I). The latter acts as an internal slice consumer and performs a series of slice requests to the slice generator unit in the same manner as described earlier. In this context however, the slice query specifically requires slices produced to consist of naked foundation slices only, in other words foundation fragments which do not possess any semantic annotations yet. Slices returned by the slice generator unit are then stored within the *naked foundation slice repository* (J) on the local file system. The semantic analyser module subsequently processes each slice stored in the repository to produce a set of semantic meta-data annotations, which are then converted and stored in the Virtuoso triple store in the same way as for the structural fragmenter module. The semantic analyser module performs these step repeatedly until no more slices matching the slice query content requirements are found by the slice generator unit. Within the context of this research, both the fragments and meta-data stored within Slicepedia and used to generate slices are referred to as slicing data.

[4] A more detailed definition of what constitutes a SliceHit is provided in section 4.3.4.1

4.2.4 Summary

This section presented a high level overview of Slicepedia's architecture. In particular the role of each underlying components and their mutual interaction was described in detail along with the overall process involved when servicing a AHS request. The following section presents a detail description of the functioning and implementation of each individual component of the slicer.

4.3 Component Setup and Implementation

4.3.1 Introduction

While the previous section presented the overall coordination of components composing the Slicepedia pipeline, this section on the other hand provides a detailed implementation description of each component used to build the slicer[5].

4.3.2 Resource Harvester Unit

As mentioned in section 3.3.2, the process of correctly identifying relevant web pages is considered out-of-scope of this research. Hence, existing third party IR systems were used instead to implement this module. What constitutes pages relevant to AHSs reuse intentions therefore depends upon what IR component in particular is selected. Traditional IR searches or more sophisticated web-crawls using for example the OCCS, (which crawls large number of resources on the WWW and analyses topic relevancy for each resource discovered) could be used (section 2.3.5). The OCCS was unfortunately no longer in operation during the course of this research. Nevertheless, many other open-source and SaaS IR tools were available at the time Slicepedia was being implemented. Open-source tools such as Heritrix[6], Nalanda[7], Combine[8] or WebSphinx[9] offer the community the ability to perform a variety of crawling tasks (from classic crawling to focused crawling). Other solutions such as PromptCloud[10], 80Legs[11], grepsr[12] or mozenda[13] offer similar capabilities available instead as SaaS

[5]Slicepedia was run within a Virtual Machine (VM) (running Ubuntu 8.04) powered by a Dell Poweredge 1950 machine, using the Xen 3.0 virtualisation software on a CentOS 5.3 host. The VM was allocated 1GB of Random Access Memory (RAM) and eight dual quad core Intel Xeon E5420 processors running at 2.5GHz were accessible to the VM if necessary.

[6]https://webarchive.jira.com/wiki/display/Heritrix/Heritrix

[7]http://ivia.ucr.edu/projects/Nalanda/

[8]http://combine.it.lth.se/

[9]http://www.cs.cmu.edu/ rcm/websphinx/

[10]http://promptcloud.com/

[11]http://www.80legs.com/

[12]http://www.grepsr.com/

[13]http://www.mozenda.com/

solutions. As the use of SaaS services requires zero infrastructure, guarantees a level of service and at the same time eases overall integration of components, the use of a SaaS solution was preferred over an in-house open-source tool deployment approach. TDR 1 specified that Slicepedia should prioritise the use of free solutions. Among the four options presented above, at the time Slicepedia was been implemented, 80Legs was the only service offering an unlimited free plan. This service additionally also offers a Java based Application Programming Interface (API) to its services. Since the slicer prototype was implemented for this research using Java, this API would also enable this crawler to be configured and run automatically within the slicer pipeline without any human interaction necessary (DR 2). For all these reasons, this service was hence selected as the preferred IR component for the slicer prototype.

The native resource identifier module (A) part of the resource harvester unit, therefore simply consists of a Java wrapper component around the 80Legs API. It configures and runs the crawler and finally forwards links discovered by this service to the URI harvester, which downloads each resource within the native resource repository (C).

4.3.3 Analyser Unit

4.3.3.1 Triple Store Meta-data and Fragment Storage

Among the range of triplestores available when the Slicepedia implementation was being carried out, three prominent triplestore solutions were considered as possible candidates; namely i) Mimir from GATE[14], ii) Aduna's Sesame[15] and iii) Openlink's Virtuoso[16]. These triplestores are similar in the sense that each of them claims to provide large and scalable RDF storage solutions. While the first two solutions achieve this objective by focusing specifically upon the storage of RDF data, Virtuoso achieves the same goal through a multi-model data server. This decision enables Virtuoso to provide a platform agnostic solution for data management, access, and integration, meaning data stored as RDF can also be accessed using traditional database queries.

[14]http://gate.ac.uk/mimir/

[15]http://www.openrdf.org/

[16]http://www.openlinksw.com/

In addition to SPARQL queries, all three solutions can also provide full-text searches. Full-text searches allows triplestore clients to query RDF triples based on keywords contained within selected RDF objects. Mimir and Virtuoso provide this functionality integrated directly within their main product, whereas Sesame requires additional components, namely the Nepomuk LuceneSail[17], to be set-up to achieve this purpose. In all three cases, these full-text search capabilities augment and combine standard SPARQL queries with traditional IR keyword-based searches within SPARQL statements.

Four important requirements constrained the selection of the triplestore used within this implementation. Since the ultimate goal of Slicepedia is to provide large volumes of resources to AHSs, the ability of the triplestore to provide *i*) a scalable, *ii*) stable and easy-to-use storage solution for this pipeline implementation was a prerequisite. Additionally, TDR 4.a and TDR 1 require Slicepedia to support *iii*) keyword-based searches and also to *iv*) prioritise the selection of open-source components.

At the time triplestore were being considered for this implementation, Mimir had only been released recently. For this reason, although it claimed to provide a scalable solution for the storage of large sets of data, no independent evaluation had been carried out at this stage and very little documentation concerning installation and usage were available. Hence, this option was considered unsatisfactory. Virtuoso and Sesame, on the other hand, are used extensively within the linked data community to store very large sets of data (*i*). However, although both are available as open-source software, the terms and conditions under which Sesame is distributed are less restrictive (*iv*). Additionally, as the installation process and usability involved with Sesame appeared much simpler, Sesame was originally selected and used for the purpose of this implementation. However, when full-text search needed to be added, only very little documentation and unmaintained software was available. For this reason, the Sesame triplestore was later switched to a Virtuoso solution. Fortunately, this required very little changes in code since both triplestore support the OpenRDF API[18] (section 5.3.5). Hence, unless specified the triplestore selected for the Slicepedia

[17] http://dev.nepomuk.semanticdesktop.org/wiki/LuceneSail

[18] http://www.openrdf.org/

implementation carried out for this research consists of the Virtuoso triplestore.

4.3.3.2 RDF Converter Module

As described in figure 4.1, slice data is stored within the triplestore through a RDF converter module (E). This module, implemented specifically for the purpose of this slicer, simply converts POJO objects, representing either fragments or annotations (section 4.3.3.4.3 and section 4.3.3.5.2) into sets of triples according to a Slicepedia RDF schema. In order to avail of full-text search combined with SPARQL capabilities of triplestores selected, both annotations produced by the analyser unit as well as foundation fragments are stored as RDF object values. This decision also enabled both types of data to be accessed using the same mechanism (simplifying implementation implications), and contributed to supporting design requirement DR 5 regarding third party slicing collaboration.

4.3.3.3 NLP Framework Selection

As specified in TDR 8, Slicepedia should rely extensively upon NLP algorithms, combined in the form of a pipeline architecture. The pipeline architecture referred to within the context of NLP algorithms should not be confused with the general slicer pipeline architecture, which performs the harvesting, fragmentation, annotation and delivery of open-corpus resources into slices. Within the context of NLP algorithms, the pipeline refers to the succession of NLP algorithms (such as tokenizer, sentence splitters etc.) used within the structural fragmenter (D) and semantic analyser (I) modules.

Among the set of NLP frameworks available at the time this implementation was being carried out, Languageware[19] developed by IBM and GATE[20] developed by the University of Sheffield were the most popular choices. Both frameworks are implemented in Java and provide the ability to structure NLP algorithms into a pipeline architecture. Each framework comes available with a set of off-the-shelf NLP algorithms that can be used directly within the respective pipelines.

[19]http://www-01.ibm.com/software/globalization/topics/languageware/

[20]gate.ac.uk

Both frameworks however, differ with respect to the end-users targeted by these systems. While GATE is primarily focused in serving the needs for programmers, Languageware aims at making text analysis accessible to non-programmers (such as humanities researchers) through the use of a no-coding, drag and drop procedure philosophy. The result of this "easy-to-use" approach however means that programmatically extending the framework can result in a very laborious process [Sweetnam2011]. Features such as conditional pipelines[21] for example, were not available in Languageware at the time this implementation was being carried out. For this reason, along with the fact that the community and set of off-the-shelf NLP algorithms available for GATE appeared larger, it was decided to implement NLP pipelines used within Slicepedia using the GATE framework. Additionally, GATE has the advantage of being completely free and open-source. Selecting this NLP framework, hence also contributed in supporting TDR 1 requiring components selected for this implementation to prioritise open-source software when possible.

[21] A more detailed overview of GATE is provided in section 2.4.4.5

4.3.3.4 Structural Fragmenter Module

This section presents the implementation of the structural fragmenter module (D) presented in section 4.2, which carries out an important step of the slicer pipeline developed for the purpose of this research. As presented in figure 4.2, the structural fragmenter module consists of a real-time GATE controller (O), executed and accessed by the rest of the slicer using the GATE embedded API (section 4.3.3.3). Whenever resources are available within the *native resource repository* module (C), this module triggers its internal NLP pipeline (O) and performs a structural analysis of these resources. Once the NLP analysis is performed, a *fragment extraction* module (P) processes the annotation output and forwards it to the RDF converter (E) for storage. The following sections describe in more details how each step is carried out.

4.3.3.4.1 Fragmentation Approach Selection

Section 3.3.3 presented the set of ideal characteristics necessary for the algorithm to be selected to perform the fragmentation process. This set of requirements are repeated below for the readers convenience in table 4.1.

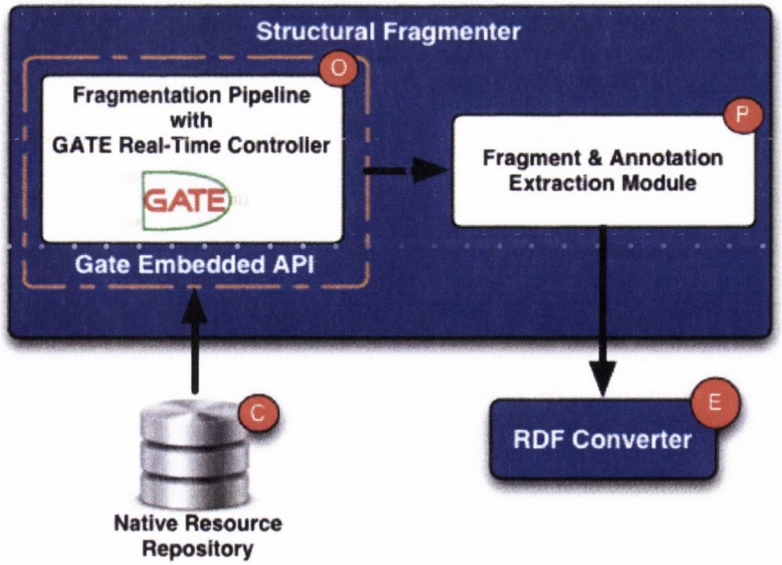


Figure 4.2: Structural Fragmenter Module

ID		Technical Design Requirements
3		Fragmenter characteristics:
	a	high speed and fully-automated processing of large volumes of pages
	b	domain and language independent
	c	content type agnostic
	d	page-level fragmenter
	e	tag meaning independent
	f	tag structure independent
	g	predictable fragment sizing control

Table 4.1: Extract from Technical Design Requirements

As web pages have evolved over time, content fragmentation has become an increasingly difficult task to perform. Besides the wide variety of pages available, each individual page itself now contains an increasing set of heterogeneous elements such as user comments, advertisements, snippets previews etc (section 2.4.3). Among the range of fragmentation algorithms reviewed within the content fragmentation state-of-the-art section of this document (section 2.4.3), the feature-based densitometric approach to fragmentation, proposed by [Kohlschutter2008a], appeared to be the most promising structural fragmentation algorithm to use for the purpose of this research. This page-level (TDR 3.d) and fully-automated approach to fragmentation, could theoretically permit large volumes of pages to be processed at high-speed (TDR 3), without any human intervention and without the need to harvest additional pages for training purposes. Since it does not depend upon any a rendering or machine learning process, this should significantly reduce any computational costs and hence increase time performance even more. Its ability to fragment documents regardless of tag meaning or structure (TDR 3.e, TDR 3.f), or without taking into account the semantics of words contained within documents should theoretically also enable a wide diversity of pages to be processed, regardless of language, domain or content type (TDR 3.b, TDR 3.c). This aspect would also benefit technical

design requirement TDR 2, which requires components used for the Slicepedia implementation to be selected based upon their multilingual support. Finally, the ability to control the size of fragments (TDR 3.g) would also appear achievable by modifying thresholds, used by the algorithm, depending on the size of fragments desired. For all these reasons, densitometric page fragmentation appeared to represent the most promising approach encountered in the state-of-the-art. However, since this algorithm relies upon tag-based documents, selecting this approach will require narrowing down the set of open-corpus resources available on the WWW to pages of obtainable in HTML format. As this format, represents the most widely used choice on the WWW, this limitation is considered tolerable for the purpose of this initial slicer implementation prototype.

As this fragmentation approach is still relatively novel, at the time this investigation was carried out, it appeared to the authors knowledge that a full evaluation of this approach and identification of strengths and weaknesses across the range of characteristics, critical for content slicing, hadn't been performed. Since fragmentation represents a decisive step of content slicing systems, an experiment evaluating the performance of densitometric fragmentation algorithms, with respect to a range of characteristics, unknown at the time this research was being carried out, is presented in section 5.2. The evaluation confirmed the suitability of this algorithm with respect to specific slicing requirements such as its ability to fragment without the need for interpreting the meaning or structure of tags within pages (TDR 3.e, TDR 3.f). Results additionally suggest this approach provides the ability to fragment pages across languages with a predictable control of both fragment sizing and cross-language accuracies (TDR 3.g, TDR 3.b). However the densitometric approach to fragmentation was discovered to be highly content type dependent (TDR 3), with higher accuracies achieved for news and encyclopaedia contents. A significant time performance decrease (TDR 3) with respect to various fragment sizing parameters was also discovered. With respect to the latter, and as discussed in section 4.4, since DR 5 requires Slicepedia to support an open and shareable approach to slicing, the decision was taken to select an arbitrary size for fragment outputs and instead control the sizing of slices within the slice generator module. For

this reason, the structural fragmenter used for the purpose of this implementation would be exempt from drawbacks related to poor time performance occurring for various sizing parameters and would instead only use a fixed set of parameters to produce fragments. With respect to content dependency, on the other hand, since Slicepedia is required to process open-corpus content (DR 1) using a content/AHS agnostic approach (DR 2), this means web-resources of any type could be targeted for slicing. Hence, the content type dependency discovered while evaluating the densitometric approach to fragmentation would clearly affect the overall performance of the slicer implemented for the purpose of this research. For this reason, it was decided to limit the range of open-corpus material on the WWW, targeted for the purpose of this research, to any html web-page originating from pages containing articles such as news, encyclopaedia or tutorial websites. Restricting the range of targeted open-corpus material available on the WWW to this set, does not interfere with the original motivation of this research consisting in processing large volumes of undetermined open-corpus resources. However, the author acknowledges that this represents a compromise on the slicer content/AHS agnosticity. This open-corpus subset still represents an immense source of content to be reused by AHSs, with a diversity sufficiently large enough to require a content/AHS agnostic slicing approach to be developed. Nevertheless, if this approach to content fragmentation is to be selected permanently for the purpose of open-corpus slicing, improvements with respect to its content type dependency will need to be achieved (see section 6.3).

4.3.3.4.2 Gate Fragmentation Pipeline

Following the selection of a fragmentation approach, the underlying set of NLP algorithms (figure 4.2), used to support the operation of the *structural fragmenter* module, were assembled using a GATE real-time controller (see section 4.3.3.3).

As in most NLP systems, and since the equation underlying DCF algorithms relies partially upon the total amount of tokens contained within a specific tag (section 2.4.3.5), the first PR contained within this controller (described in table 5.3.5) consists of a ***tokenizer*** (see section 2.4.4) provided as part of the ANNIE set of GATE NLP

Fragmentation NLP Pipeline		
1	ANNIE English Tokenizer	
2	ANNIE Sentence Splitter	
3	**BoilerPipe**	
	a	Densitometric Fragmenter
	b	Boilerplate Detector
4	Fragment Sequencer	
5	Annotation Transfer	
6	**Internal Fragment Pipeline**	
	a	Fragment Type Identifier

Table 4.2: Fragmentation Pipeline

algorithms . The output of this tokenizer PR consists of annotations pointing to the start and end of each token identified within the content processed.

Following the tokenizer, is the ANNIE english ***sentence splitter***. This domain and application-independent PR consists of a cascading set of finite-state transducers, which use the token annotations produced by the previous PR to identify sentence boundaries within the text. A list of abbreviations are used to help distinguish sentence boundaries marked by a full stop from others. Each sentence identified is assigned an annotation of type "sentence" (pointing to the start and end of each sentence).

Directly following the tokenizer is the ***fragmentation algorithm***. As explained previously, an evaluation analysing the performance of the DCF approach was undertaken for the purpose of this research (section 5.2). At the time this evaluation was carried out, no implementation of the original algorithm presented in [Kohlschutter2008a] was available. Hence, several variations of this algorithm were implemented to pursue this task. Nevertheless, by the time these evaluations were accomplished and the Slicepedia pipeline was in development, the original authors of this approach did eventually release their implementation as open-source software

(also available as a GATE PR). Since this DCF implementation, named Boilerpipe[22], had undergone thorough bug testing prior to release, it was decided to select this more stable implementation of the algorithm as opposed to the one implemented in-house. This decision also supports design requirement TDR 1 which requires the Slicepedia implementation to make extensive use of open-source software, as well as enabling other researchers within the community to reproduce the experiments carried out as part of this research using freely available software.

Finally, Boilerpipe also includes a ***boilerplate detection algorithm*** (section 2.4.4.4) using shallow text features of documents [Kohlschutter2010a]. As mentioned previously, the fragmentation of native resources also requires the ability to distinguish fragments based upon whether they consist of menus, advertisement etc. Hence, the use of a stable implementation integrating both densitometric fragmentation with a boilerplate detection algorithm was considered a better decision, as opposed to using an in-house approach lacking boilerplate detection. This approach to boilerplate detection achieves very similar performance to more complex alternatives at much lower computational costs [Pomikalek2011] [23]. Boilerplate detection using shallow text features shares in essence, many similarities with the densitometric approach to fragmentation. It is content type and domain independent and does not require any inter-document knowledge, which means pages can be processed independently of each other [Kohlschutter2010a].

At the time the Slicepedia implementation was being carried out, the performance of this approach across languages was unknown. Since this approach relies primarily upon word count (treated as tokens) within tags, as for DCF algorithms, the authors assumption was that this approach would perform similarly as its fragmentation counterpart. Nevertheless, if Slicepedia is to be extended for multilingual slicing, the performance of this approach to boilerplate detection with respect various languages should be evaluated (section 6.3).

The PR encapsulating Boilerpipe uses the token annotations produced by the

[22] http://code.google.com/p/boilerpipe/

[23] http://tomazkovacic.com/blog/122/evaluating-text-extraction-algorithms/ [Accessed: October 3, 2014]

tokenizer to count tokens contained within each tag of the original documents and calculate densitometric values as described in section 5.2. It produces two types of annotations each assigned to fragments produced from the original document and selected based upon whether it considers these to be boilerplate content or not.

A ***fragment sequencer*** PR consisting of JAPE grammar rules, written specifically for the purpose of this slicer, subsequently identifies each non-boilerplate fragment as a valid foundation fragment and assigns sequence numbers to each one based upon their position within the original native resource. These sequence numbers are later on used by the *slice generator* unit to construct slices. During this phase, compound fragments are also identified. Compound fragments consists of a sequence of foundation fragments uninterrupted by boilerplate content. Whenever the fragment sequencer encounters a foundation fragment identified as boilerplate, a new compound fragment is created, containing the next non boilerplate foundation fragment encountered. The use of compound fragments is based upon the assumption that content held within native resources, uninterrupted by boilerplate content, must possess some degree of structural coherence and unity. Compound fragments hence, allow the slicer to preserve this original structural coherence of the resources that have been sliced based upon the original structural position of fragments in native resources. As for foundation fragments, compound fragments are also assigned compound sequence numbers. Since GATE treats original mark-up tags contained in HTML pages as annotations (section 4.3.3.3), once each valid foundation fragment is identified and sequenced, the GATE ***annotation transfer*** PR identifies each original mark-up annotations contained within valid fragments, as well as those produced within this pipeline, and saves them within a Slicepedia annotation category.

Finally, a set of JAPE grammar rules is used to produce fragment specific annotations based upon the annotations transferred previously. These grammar rules are run within a ***Fragment type Identifier*** PR (implemented specifically for Slicepedia) as part of a ***internal GATE pipeline*** (section 4.3.3.3) for each foundation fragment encountered. Fragment type annotations for example, are produced based upon original mark-up annotations, contained within each fragment, in order to specify

whether a fragment consists of a table, paragraph, bullet points or title. Since TDR 5.a requires Slicepedia to take into account the sizing of slices requested by slice consumers, the size of each fragment (specified with respect to the number of token and sentence annotations contained within each fragment) is also determined and stored as a fragment size annotation.

For the purpose of this experiment, and since the subset of open-corpus resources targeted for this experiment are constrained to the English language as a first step (section 3.3.3), the tokenizer and sentence splitter algorithms selected for this pipeline were english specific NLP algorithms. However, many algorithms designed to achieve these specific tasks for various languages are also available [Zhang2007, Branco2009]. In the event, additional languages were targeted for slicing, these alternative tokenizers could easily be selected using a conditional controller (as mentioned in section 4.3.3.3) based upon the output of language identifier PRs[24].

4.3.3.4.3 Fragment Extraction

Once native resources are processed by the GATE fragmentation pipeline, the *fragment extraction* module (P) extracts annotations produced along with the text pertaining to each fragment, using the GATE embedded API. Mark-up annotations originally contained within each foundation fragment are reinserted into the text as they originally appeared within native resources. Annotations related to each foundation fragment are associated with text containing original mark-up to produce foundation fragment POJO objects, which are subsequently forwarded to the *RDF converter* module (E) for storage. During this phase, compound fragment POJO objects are also created to represent each compound fragment identified by the fragment sequencer. Finally, the fragment extraction module also creates one additional page-level compound fragment representing all non boilerplate fragments extracted from each native resource.

[24]http://gate.ac.uk/sale/tao/splitch15.html

4.3.3.5 Semantic Analyser Module

The set of analysis performed by the *semantic analyser* module is strongly determined upon the range of content requirements potentially requested by slice consumers. Although specific content requirements of individual slice consumers are unpredictable, an estimation of very broad range of needs potentially needed by a group of slice consumers can be determined. Hence, the set of NLP algorithms used for this implementation were selected based upon additional design requirements which Slicepedia should support, as well as various slice consumer needs used for the evaluation carried out in the following chapter. The set of NLP algorithms, converted into Processing Resources (PRs) and combined within a GATE real-time controller is presented in table 4.3.

Within the context of this research, the concept referred to as *semantic analysis*[25], which is performed by this module, is used in the broadest possible sense of the term. It is used in particular in opposition to the notion of *structural analysis* (performed by the *structural fragmenter*), which only focuses upon the structure of open-corpus resources.

4.3.3.5.1 Gate Semantic Analysis Pipeline

As mentioned in section 2.4.4, most NLP pipelines start with a similar set of standard algorithms and this pipeline is no exception. The first two PRs used in this pipeline consists of the same ***tokenizer and sentence splitter*** used within the *structural fragmenter* module (section 4.3.3.4).

The two pipelines however differ from the third PR onwards. This pipeline uses instead a ***Part-Of-Speech (POS) tagger*** [Hepple2000] to assign a POS annotation to each token encountered within the text. This algorithm is partially based upon the approach proposed by [Brill1992] and uses a ruleset derived from training results

[25]Semantic analysis here refers to both traditional synthetic analysis (POS, morphological analysis etc.) and/or Information Extraction (IE) algorithms (named entity recognition, relationship extraction).

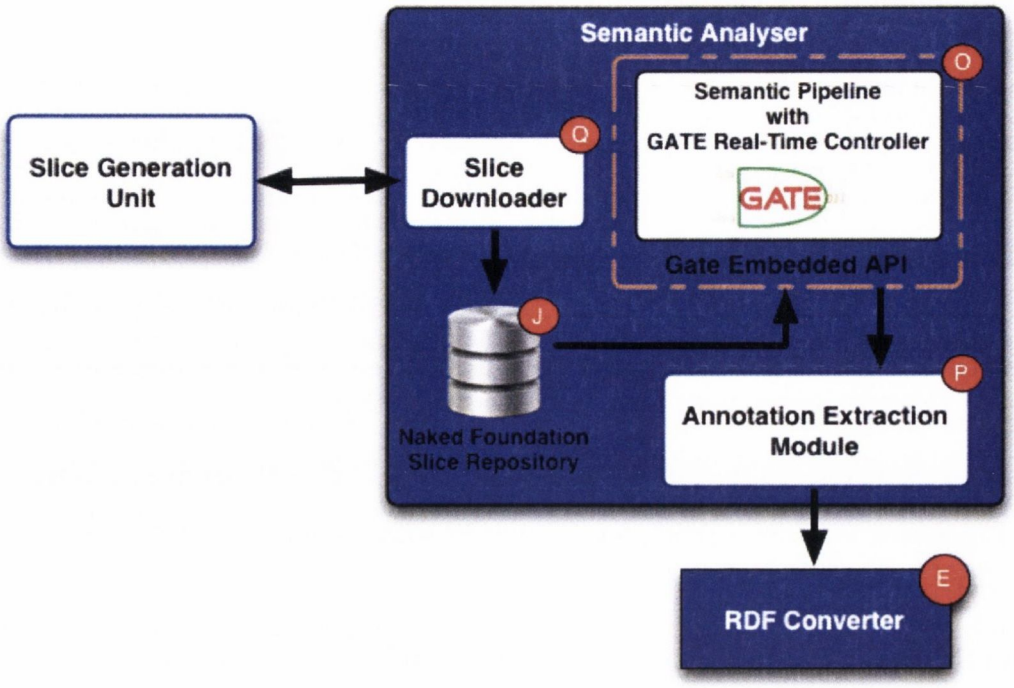


Figure 4.3: Semantic Analyser Module

performed upon a corpus taken from the Wall Street Journal[26].

The next three PRs consist of i) a morphological analyser, ii) a verb group chunker and iii) a verb feature analyser[27]. These three algorithms were selected in order to comply with TDR 5.c, which requires Slicepedia to support AHSs page adaptation. As will be described in section 5.4.2.4, the slice consumer used to perform this adaptation consisted of a Language Web-Training tool presenting traditional English grammar exercises to students. Page adaptation carried out by this AHS is performed with respect to verbs contained within resources delivered by the slicer. For this reason, Slicepedia should be able to take into account AHS content requirements specifying various attributes of slices to be delivered based upon verb features contained within these resources. Lemmas (section 2.4.4) identified by the ***morphological analyser*** PR, are hence added to the set of features contained by each token referred to. LRs are subsequently processed using the domain independent ***verb group chunker*** in

[26] www.wsj.com

[27] Both the morphological analyser and verb group chunker were developed by GATE using JAPE rules applied upon the token and POS annotations

order to produce annotations of type *Verb Group* with verb features detected such as type, tense, voice, neg, etc. Finally, once the morphological analyser and verb group chunk have be executed upon the LR, the ***verb feature analyser*** PR, developed as a JAPE transducer specifically for the purpose of this research, produces the non-finite form of each verb group identified by the verb group chunker[28]. The set of rules written for this transducer, take into account the lemmas produced earlier as well as each feature contained within the *Verb Group* annotation. For each verb group analysed this PR adds the non-finite as well as original form of the verb chunk as encountered within the text, as annotation features within the *Verb Group* annotation.

DR 4 and TDR 7, requires slices to be assigned domain anchor points, in the form of linked data concepts, so as to be more easily integrated within third party AHSs. Additionally, TDR 5.a also requires Slicepedia to provide the ability to focus native resources upon concepts selected by slice consumers. For these reasons, the semantic analysis pipeline includes a concept tagging algorithm (section 2.4.4.4) which assigns linked data concepts to regions of naked slices analysed. While traditional text segmentation algorithms could have been used to semantically segment pages, using such an approach would have required slices to be semantically segmented a-priori of slice consumer requests on arbitrary concepts (hence failing to support TDR 5). For this reason, an approach similar to that proposed by [Leoncini2012] was instead selected for this prototype in order to fulfil both of these design requirements. Hence, Slicepedia uses the AlchemyAPI service[29], to assign conceptual annotations to slices based upon the topics covered in each slice. These annotations refer to unique Dbpedia[30] linked-data concepts and are assigned a relevance value (ranging from 0 to 1) based upon the algorithm's degree of certainty. Each annotation produced by the AlchemyAPI also contains the start and end location of regions of pages covered by each concept. Since Dbpedia concepts are derived from Wikipedia, the range of

[28]The non-finite form of a verb chunk consists in a succession of several verbs in a form which is not inflected for tense or other grammatical categories as well (i.e. not for gender, person, number, aspect, mood, voice)

[29]http://www.alchemyapi.com/

[30]www.dbpedia.org

concepts covered by this openly available linked-data repository is very large. For this reason, concept annotations produced by this algorithm are used as domain anchor points for AHS (TDR 7). The selection of this linked-data repository hence enables Slicepedia to cover the largest possible range of domains, which could be of use to arbitrary slice consumers. Also since Dbpedia offers hierarchies of concepts, these Dbpedia annotations can be used by the slice creator module (H) to focus slices on demand (TDR 5.a) based upon arbitrary concepts provided by slice consumers using this hierarchy (section 4.3.4.4). For the same reasons as explained when selecting the web-crawler used for this prototype (section 4.3.2), an approach using a free SaaS concept tagging service as opposed to the deployment of an open-source solution in-house was preferred. Since no GATE PR was available for the AlchemyAPI service, a PR running and extracting annotations produced by this service was created for the purpose of this prototype.

Finally, the NLP pipeline used within this module, also includes an analysis of the ***reading difficulty*** of resources processed. This PR, implemented specifically for the purpose of this research using the Flesh Java library[31], assigns a *Flesch Reading Ease* and *Flesch-Kincaid Reading* score annotation to each LR processed. Both scores use the same core measures (word length and sentence length) with different weighting factors to calculate ease of reading. The *Flesch Reading Ease* score consist of a number between 0 and 100, with lower values requiring a higher level. Reading scores lower than 30 for example are best understood by university graduates, while resources assigned a score between 90 and 100 can be understood 11 year old students. The *Flesch-Kincaid Reading* score translates the 0100 score to a U.S. grade level, making it easier to judge the readability level of various books and texts.

As with the structural fragmenter module, if a multilingual set of open-corpus resources is to be targeted by the slicer, in addition to the tokenizer and sentence splitter PRs, various versions of the POS tagger, morphological analyser and verb group chunker PRs would need to be included within the pipeline for each language supported. As for the tokenizer and sentence splitter, many of these algorithms are

[31] http://flesh.sourceforge.net/

Annotation NLP Pipeline	
1	ANNIE English Tokenizer
2	ANNIE Sentence Splitter
3	Part-Of-Speech Tagger
4	GATE Morphological Analyser
5	Verb Group Chunker
6	Verb Feature Analyser
7	Concept Annotator
8	Reading Level Analyser

Table 4.3: Semantic Analysis Pipeline

available for different languages [Schmid1995].

4.3.3.5.2 Annotation Extraction

In a similar manner as with the fragmentation module (section 4.3.3.4.3), once the GATE pipeline is run, the annotation extraction module (P) extracts annotations produced using the GATE embedded API and converts them into POJO objects, which are then transferred to the RDF converter module (E) for storage.

4.3.4 Slice Generation Unit

4.3.4.1 Slice Generation Overview

The procedure carried out to produce slices consists of three steps presented in figure 4.4. As described in section 4.2.3, the slice generation unit is composed of two submodules, each responsible for the first and third step respectively. The first step carried out by the *slice search module* (G) (section 4.3.4.3) searches for possible slices to be generated, matching arbitrary content requirements provided by slice consumers using a SliceQuery object (section 4.3.4.2). The slice search module returns a list of slice hits to the slice consumer, which describe the specifications of each possible slice candidate available.

A SliceHit object consists of a blueprint for slices. It contains a description of all the attributes and information necessary (fragments and meta-data) for Slicepedia to construct a slice. It can be stored by slice consumers and later on be forwarded back to Slicepedia to convert it into slices. A sliceHit would typically contain lists of URIs pointing towards individual fragments required to build an arbitrary slice. It also contains attributes (such as size of slice, content style etc.) describing the right-fitting characteristics of the corresponding fictitious slice it describes. The type of attributes returned within a slice hit match those specified within the SliceQuery. If requested such attributes can also contain URIs pointing to DBpedia concepts covered by the slice (section 4.3.3.5). During step two of the procedure, SliceHits can be modified by slice consumers according to individual content needs. The purpose of this intermediate step, providing a level of indirection, is to offer as much control as possible to the slice consumer over the production of slices by Slicepedia (DR 6). The use of SliceHits enables slice consumers to i) control what slices are ultimately produced by Slicepedia and also ii) compare possible available options before being supplied content. If the slice consumer does not require such a level of control, it simply leaves the sliceHits unmodified and proceeds to step three. In most cases however, SliceHits will need to be modified in order to better match the content requirement needs of individual slice consumers. In these circumstances,

a slice consumer can analyse, select and/or modify arbitrary attributes of slices it wishes to retain. If slices are to be focused on specific concepts for example, the conceptual hierarchy related to DBpedia concepts available within each slice hit, can be analysed by the slice consumer, through the SPARQL DBpedia endpoint[32] in order to determine which concepts pertain to the arbitrary scope desired. Concepts determined not to be within that scope can then be removed from the SliceHit by the slice consumer and will be ignored by the slicer. Those falling within the scope desired can subsequently be specified as concepts for slices to be focused on within a new slice search step (section 4.3.4.4). If a slice consumer is not satisfied with the SliceHits returned, it can return to step one and request new SliceHits with different content requirements.

If a number of Slicehits received (and possibly modified) by the slice consumer do possess a description which appears to be adequate for the intended reuse purpose, the latter forwards the arbitrary chosen SliceHits to the *slice creator module* (H), which constructs the corresponding slices based upon specifications provided in each Slicehit. The next three sections, explain in more details the implementation carried out for the SliceQuery, slice search and slice creator module.

[32]http://dbpedia.org/sparql

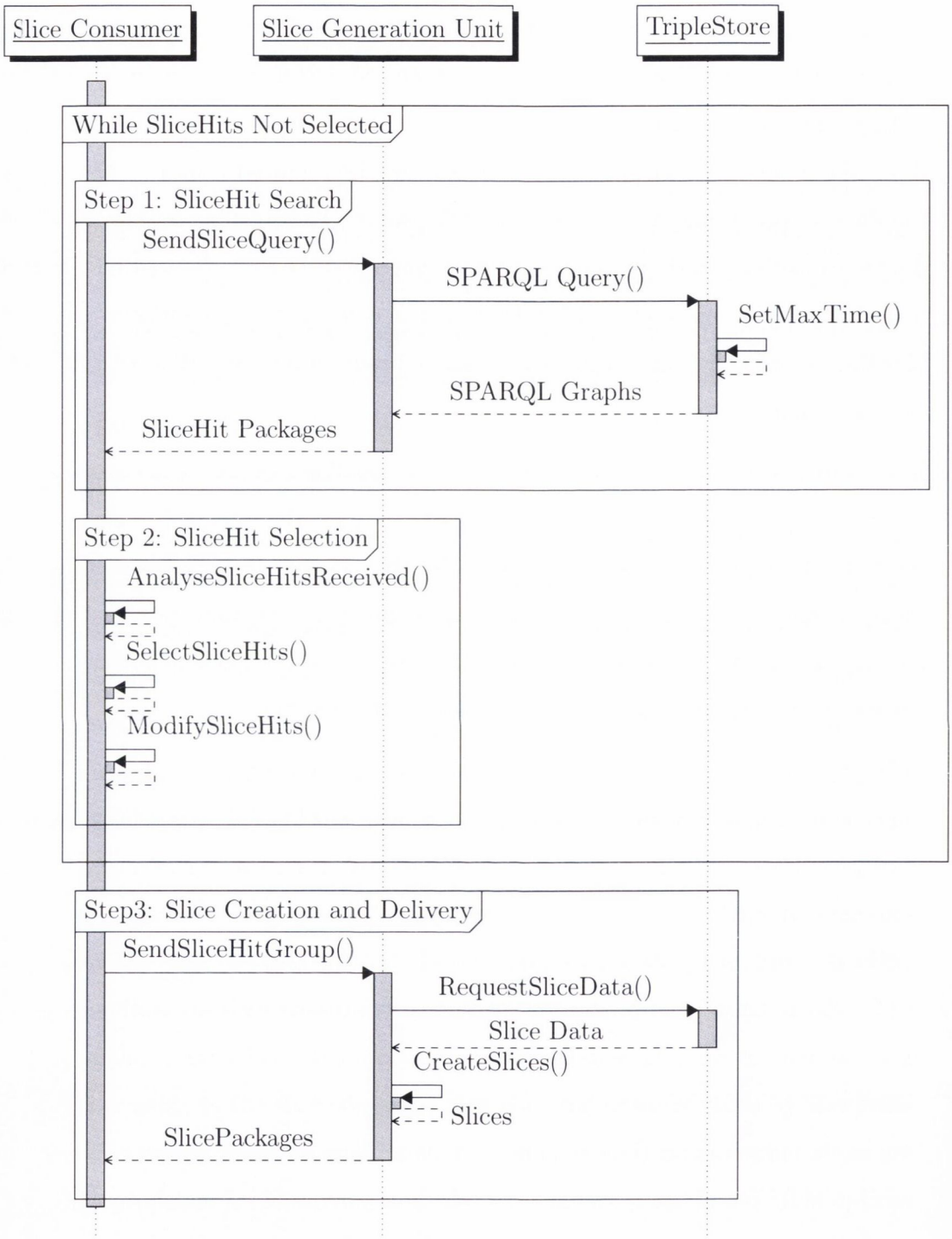


Figure 4.4: Slice Search Procedure

4.3.4.2 SliceQuery Object

Although slice requests can be performed directly using SPARQL queries through the Slicepedia SPARQL interface (section 4.5), it was decided to provide an abstraction layer encapsulating niche content requirements of slice consumers. This decision enables a separation of concerns between how slice consumers search for slices and how a slicer represents slice data internally. As described in section 3.3.3, due to DR 5, which requires Slicepedia to provide a mean to share slicing data with independent institutions, RDF was selected for the internal representation of slice data for the purpose of this prototype (TDR 7). Nevertheless, there is no reason why such data could not be presented using alternative technologies such as relational databases. Additionally, encapsulating slice requests using a POJO also enables such requests to be easily converted to a json format and transmitted between slice consumers and slicers using a REST API (section 4.5), hence also supporting TDR 6.b. Finally, as can be seen in figure 4.6, SPARQL queries can very rapidly become quite complex. Representing slice queries in this manner simplifies the representation of such queries significantly.

A Unified Modeling Language (UML) representation of the SliceQuery object available to Slicepedia slice consumers is presented in figure 4.5. Methods available to define slice requests content requirements are divided into five categories; namely 1) Structural, 2) Semantic, 3) Granularity, 4) Ordering and 5) Miscellaneous specifications. All the methods available within SliceQuery are non-exclusive, hence all of these can be used together in multiple combinations depending upon the content requirement complexity of slice consumers.

Structural specifications can be defined through the use of six methods. Each enable slice consumers to specify a number of (including none) of tables, bullet lists, titles, code or forms, which slices returned should possess. The ability to specify whether slices should contain plain text or not is also available.

Semantic specifications are available through five methods. As mentioned in section 4.3.3.5, the set of semantic specifications available to slice consumers is directly related to the type of semantic analysis selected by slicer designers. With respect to

this implementation, reading difficulty, concepts covered, verb chunk attributes and keyword searches were selected for the purpose of this research. The *keywordSearch* method enables a slice consumer to provide a String object containing a list of keywords (as well as standard IR operators if needed), which slices retrieved should possess. The i) *isAbout*, ii) *hasVerbWithTense* and iii) *hasReadingDifficulty* let slice consumers respectively provide either i) a set of DBpedia URIs corresponding to the concepts slices should cover along with their minimum concept relevancy, ii) the tense some verbs within slices should be conjugated at, as well iii) Flesch/FleschKincaid reading scores.

Slice Granularity delivered by Slicepedia can be specified by providing an acceptable range of words, sentences or number of paragraphs which could satisfy sizing criteria of a slice consumer. The ability to require slices to be as large as possible (i.e: consisting of all available fragments originating from a native resource) is also provided through the *setMaxSize* method. Finally, SliceQuery also enables slice consumers to specify whether slices should be focused upon Dbpedia concepts (using the *isAbout* method) or annotations requested (using the *focusOnAnnotations* method). As section 4.3.4.4 will describe, the *isAbout* method additionally provides the ability for slice consumers to request focused slices, without removing content not covering a selected set of DBpedia concept, but instead to annotate content which covers those concepts.

Slice Ordering can also be controlled by slice consumers using four methods specifying whether slices should be ordered based on their size, DBpedia relevance, keyword-search scores, or reading score. These four options are non-exclusive and can be combined together.

Slicepedia also offers slice consumers ***additional control*** over content delivered such as *slice delivery paging* (using *setSliceStart* and *setMaxSliceNumber*) or the ability to select a list of arbitrary pages from which slices should be produced (using *hasOriginalSourcePageURIs*). Whether hyperlinks contained within native resources are returned within slices can also be specified along with the removal or replacement of HTML tags with arbitrary content (*removeUrls*, *replaceTagContent*

and *setTagsToRemove*).

As mentioned in section 4.4, slice consumers can also specify the meta-data set, originating from selected trusted institutions, to be used when producing slices by using the *setGraphContext* method, which requires as a parameter, a list of context graph URIs assigned to each institution (to support design requirement DR 5).

Finally, since SPARQL queries return every single possible RDF graph combination encountered in a triplestore, SPARQL result binding set objects returned by triplestores will generally contain various SliceHits originating from the same native open-corpus resource (with different granularity options for example). For this reason, Slicepedia additionally provides the ability to ***group results***, retrieved from the triplestore, based upon the origin of SliceHits (using the *setUniqueSourceSlices* method). This feature is very useful when performing a slice search and slice create request together. Whenever this feature is specified, each SliceHit or Slice delivered to slice consumers will originate from a unique native open-corpus resource (similarly to traditional IR searches). Slicepedia converts this specification into SPARQL using the GROUP BY clause (available in the SPARQL 1.1 specification[33]), which randomly selects one single RDF binding among all of those contained within a group. By default, Slicepedia will group results by open-corpus origin (using *setUniqueSourceSlices* method) and specify the sizing of slices to be larger than one word. The number of bullet-point list and tables are set to zero and result bindings are ordered by decreasing number of annotations (if any annotations were requested).

Slice Request Example: Figure 4.6 depicts an example a slice request represented as a slice query java object and SPARQL query. The following query consists of a request for slices similar to those performed during the Slicegap experiment presented in section 5.4. An example of the a slice returned by this query is presented in section 4.3.4.4. *Slices should be i) about the dbpedia topic "united nations" ii) with a relevance above 7.8. They should iii) not contain any bullet point lists iv) or tables, and v) should mention the words "chemical attack". The vi) size of slices should consist of at least 100 words and vii) should be focused upon the section*

[33]http://www.w3.org/TR/2013/REC-sparql11-query-20130321/

referring to the DBpedia topic requested. The viii) reading level of slices returned should be targeted to advanced English readers (below 50 on the Flesh index) and ix) all verbs conjugated in the present tense should be annotated within the text, and associated with their corresponding lemma. Finally, x) a maximum of 10 slices should be returned, xi) ordered by reading difficulty and DBpedia relevance.

SliceQuery
Structural
hasNumberOfTables(int tableNumber, Sign value) : void
hasNumberOfBulletLists(int bulletListNumber, Sign value) : void
hasNumberOfTitle(int number, Sign value) : void
hasNumberOfCode(int number, Sign value) : void
hasNumberOfForms(int number, Sign value) : void
hasPlainText(boolean value) : void
Granularity
hasNumberOfWords(int wordNumber, Sign value) : void
hasNumberOfParagraphs(int paragraphNumber, Sign value) : void
hasNumberOfSentences(int paragraphNumber, Sign value) : void
setMaxSize(boolean value) : void
focusOnAnnotations(String annotation) : void
Content Adaptation
setTagsToRemove(List<String> tagToRemove) : void
removeUrls() : void
replaceTagContent(String tagToRemove,String replacement) : void
Third Party Collaboration
setGraphContext(List<URI> newSearchContext) : void
Paging
setSliceStart(int sliceStart) : void
setMaxSliceNumber(int sliceNumber) : void
Miscellaneous
hasOriginalSourcePageURIs(List<String> uris) : void
addDefaultParameter(String featureName, String featureValue) : void
getSliceQueryId() : void
setUniqueSourceSlices(boolean uniqueSourceSlices) : void

Semantics
keywordSearch(String keywordSearch) : void
about(List<URI> concept, Double relevance, String sign, boolean focus) : void
hasVerbWithTense(String tense) : void
hasReadingDifficulty(Double value, Sign value) : void
hasReadingDifficulty(Level value) : void

Ordering
orderSlicesBySliceSize() : void
orderSlicesByDBpediaScore() : void
orderSlicesByReadingScore() : void
orderSlicesByFullTextScore() : void

Figure 4.5: SliceQuery Object

Java SliceQuery

```
SlicepediaQuery sliceQuery = new SlicepediaQuery();
sliceQuery.setMaxSliceNumber(10);

sliceQuery.paramHasNumberOfBulletLists(0, "=");
sliceQuery.paramHasNumberOfTables(0, "=");

sliceQuery.addFullTextSearchKeywordRegex("chemical attack");

sliceQuery.paramHasNumberOfWords(100, ">");

sliceQuery.paramANNOTATION_VERB_WITH_TENSE("simpre");
sliceQuery.paramANNOTATION_READING_DIFFICULTY(new Double(50), "<");

sliceQuery.paramANNOTATION_CONCEPT_FEATURE_HAS_DBPEDIA(http://dbpedia.org/resourc
e/united_nations, 7.8, ">", true);

sliceQuery.orderSlicesByReadingScore();

sliceQuery.removeUrls();
```

SPARQL Conversion of Java SliceQuery

```
SELECT DISTINCT ?sliceid ?originalSource ?FullTextSearchOrderItem ?startNode2 ?en
dNode2 ?foundationSliceAnnotated2 ?annotationType2 ( "http://dbpedia.org/resource/
united_nations" as ?dbpedia2 ) ?startNode3 ?endNode3 ?foundationSliceAnnotated3
?annotationType3 ?hasfleschreadingeasescore_slicepediaOrder ?startNode4 ?endN
ode4 ?foundationSliceAnnotated4 ?annotationType4 FROM <http://www.tcd.ie/test>
WHERE {

       ?sliceid <http://www.slicepedia.org/ontology#hasOriginalSourcePageURIs> ?c
       atId1 .
       ?catId1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#li> ?originalSource .
       { {
              ?sliceid <http://www.slicepedia.org/ontology#hasContent> ?content .
              ?content bif:contains "chemical attack" OPTION (score ?FullTextSear
              chOrderItem) .
              } } UNION { {
              ?sliceid <http://www.slicepedia.org/ontology#hasSubSlices> ?subslic
              esNode .
              ?subslicesNode <http://www.w3.org/1999/02/22-rdf-syntax-ns#li> ?sub
              slices .
              ?subslices <http://www.slicepedia.org/ontology#hasContent> ?content.
              ?content bif:contains "chemical attack" OPTION (score ?FullTextSear
              chOrderItem) . } }
```

```
?sliceid <http://www.slicepedia.org/ontology#hasNumberOfTables> ?hasNumber
OfTables3 .
FILTER (?hasNumberOfTables3 = 0) .
?sliceid <http://www.slicepedia.org/ontology#hasNumberOfBulletPoints> ?has
NumberOfBulletPoints5 .
FILTER (?hasNumberOfBulletPoints5 = 0) .
?sliceid <http://www.slicepedia.org/ontology#hasNumberOfParagraphs> ?hasNu
mberOfParagraphs6 .
FILTER (?hasNumberOfParagraphs6 > 5) .
?sliceid <http://www.slicepedia.org/ontology#hasNumberOfTokens> ?hasNumbe
rOfTokens8 .
FILTER (?hasNumberOfTokens8 > 0) .

?sliceid <http://www.slicepedia.org/ontology#hasAnnotation> ?annotation2 .
?annotation2 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?annotation
Type2 .
?annotation2 <http://www.slicepedia.org/ontology#hasNodeStart> ?startNode2.
?annotation2 <http://www.slicepedia.org/ontology#hasNodeEnd> ?endNode2 .
?annotation2 <http://www.slicepedia.org/ontology#annotatesFoundationSlice>
?foundationSliceAnnotated2 .
?annotation2 <http://www.slicepedia.org/ontology#hasrelevance> ?hasrelevan
ce1 .
FILTER (?hasrelevance1 > 0.5) .

?annotation2 <http://www.slicepedia.org/ontology#hasdbpedia> "http://dbped
ia.org/resource/united_nations" .
?sliceid <http://www.slicepedia.org/ontology#hasAnnotation> ?annotation3 .
?annotation3 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?annotation
Type3 .
?annotation3 <http://www.slicepedia.org/ontology#hasNodeStart> ?startNode3.
?annotation3 <http://www.slicepedia.org/ontology#hasNodeEnd> ?endNode3.
?annotation3 <http://www.slicepedia.org/ontology#annotatesFoundationSlice>
?foundationSliceAnnotated3 .
?annotation3 <http://www.slicepedia.org/ontology#hasfleschreadingeasescore>
?hasfleschreadingeasescore_slicepediaOrder .
?annotation3 <http://www.slicepedia.org/ontology#hasfleschreadingeasescore>
?hasfleschreadingeasescore1 .
FILTER (?hasfleschreadingeasescore1 < 50.0) .

?sliceid <http://www.slicepedia.org/ontology#hasAnnotation> ?annotation4 .
?annotation4 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?annotation
Type4 .
?annotation4 <http://www.slicepedia.org/ontology#hasNodeStart> ?startNode4.
?annotation4 <http://www.slicepedia.org/ontology#hasNodeEnd> ?endNode4 .
?annotation4 <http://www.slicepedia.org/ontology#annotatesFoundationSlice>
?foundationSliceAnnotated4 .
?annotation4 <http://www.slicepedia.org/ontology#hastense> "simpre" . }

ORDER BY DESC(?hasfleschreadingeasescore_slicepediaOrder )
DESC(?FullTextSearchOrderItem )
LIMIT 1000
```

Figure 4.6: SliceQuery SPARQL Convertion

4.3.4.3 Slice Search Module

Each slice search proceeds according to the flowchart presented in figure 4.7. Whenever the *slice search* module receives a SliceQuery object, it initially checks in its cache whether or not a triplestore SPARQL result binding set object has already been received for a slice request with the same SliceQuery ID. A triplestore SPARQL result binding set represents a list of possible RDF graph combinations which match the selected SPARQL query. If no binding is available, the SliceQuery object is converted into a SPARQL query (see previous section) and executed by the triplestore. To avoid response time hanging , an arbitrary maximum query response time is also set on the triplestore (see figure 4.4). This guarantees the triplestore will stop searching for graph combinations, matching the desired SPARQL query, when the time limit is reached. When this limit is reached, the triplestore returns the result graphs it succeeded in identifying in the time available and stores the SPARQL result binding response object it received within the *slice search* module cache. Each SPARQL binding result object returned by the triplestore corresponds to one valid SliceHit. Hence, depending upon the paging SliceQuery attributes (see previous section), the corresponding RDF graph binding matching the SliceQuery "slice start" attribute is identified, and the number of relevant RDF bindings requested are retrieved from the SPARQL binding result set and converted into SliceHits. As mentioned above, each value contained within the binding set returned by the triplestore corresponds to a SliceHit attribute. The set of SliceHits created is then finally returned to the slice consumer.

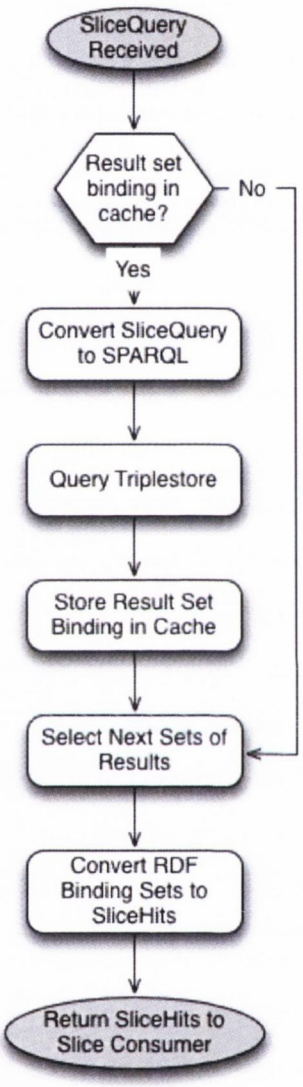


Figure 4.7: Slice Searcher Flow Chart

4.3.4.4 Slice Creator Module

As described in figure 4.8, when SliceHits are received by the *slice creator* module, fragment IDs (section 4.3.3.4) contained within each SliceHit are retrieved and used within a SPARQL query to obtain the content of each fragment from the triplestore. If fragments specified within the SliceHit consist of compounded fragments, foundation fragments referred to by this fragment are retrieved. Once all foundation fragments are retrieved, corresponding annotations arbitrarily selected by slice consumers (and specified within each SliceHit object) are queried in the triplestore and inserted (along with their requested attributes) within the original fragment content. Foundation fragments are then ordered in chronological order according to sequence numbers previously assigned by the structural fragmenter module. Content corresponding to each fragment, along with their annotations, are then merged together in chronological order into one larger unit. If the slice request required slices to be focused upon an arbitrary set of concepts Γ, only fragments annotated as covering these topics are merged together. The set of concepts to be considered during this phase are selected by the slice consumer (during the second step in figure 4.4) based upon concept hierarchies available within DBpedia (section 4.3.4.1). Whenever fragments not covering this set of DBpedia concepts Γ are about to be merged, these are replaced by an arbitrary delimiter text[34] to notify readers that original content not covering these concepts was removed. Alternatively, and if requested by slice consumers (section 4.3.4.2), content not covering this set of concepts can be merged and fragment content covering concepts Γ are annotated as such within the text, in order to inform slice consumers these parts of the text are covering concepts requested (figure 4.9). If the slice consumer requested slices to be focused on arbitrary annotations, the first and last annotation of the type specified (contained within the merged fragments) are identified. Any content (within the merged fragments) away from these two annotations by more than ζ tokens is discarded. Tags, contained within foundation fragments and specified by slice consumers to be removed or replaced, are then identified and either removed or replaced by arbitrary content. Finally, meta-data

[34]such as "[Original Text Removed]"

corresponding to the specificities (size, annotations, focused or not etc.) of the merged fragment are added to the content to form a slice as illustrated in figure 4.9.

As mentioned in section 6.3.4, multi-source slicing is reserved as future work. Nevertheless as can be noticed, the slice creator module does involve a minimal degree of recomposition driven by the original sequence order of fragments produced from native resources.

A slice output example, corresponding to slice request described in section 4.3.4.2 is presented in figure 4.9.

4.3.5 Summary

This section presented individual components involved within the slicer's architecture and how each achieves the tasks it is required to perform within the overall architecture workflow. The following sections present implementation aspects of the slicer, concerning its general usage by AHSs as well as in collaboration with third party institutions

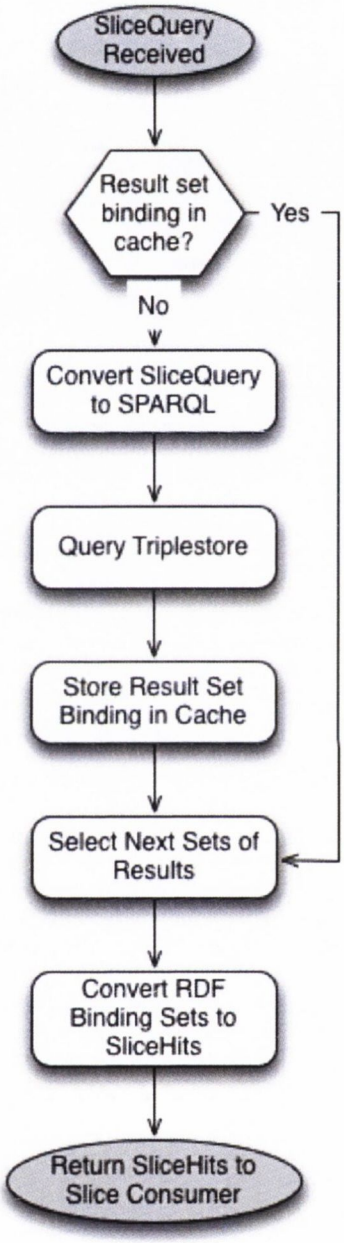


Figure 4.8: Slice Creation Procedure

```
<?xml version="1.0"?>
<slice xmlns:slicepedia="https://www.scss.tcd.ie/~levachk/slicepediaNamespace">
	<slicepedia:metadata>
		<slicepedia:source_url_meta_data>http://www.bbc.co.uk/news/world-mi
		ddle-east-24068867</slicepedia:source_url_meta_data>
		<slicepedia:url_removal>true</slicepedia:url_removal>
		<slicepedia:dbpedia>http://dbpedia.org/resource/united_nations</sli
		cepedia:dbpedia>
		<slicepedia:focused>true</slicepedia:focused>
		<slicepedia:keywords>chemical attack</slicepedia:keywords>
		<slicepedia:verbChunk>simpre</slicepedia:verbChunk>
		<slicepedia:size>148</slicepedia:size>
		<slicepedia:bulletpoint>0</slicepedia:bulletpoint>
		<slicepedia:table>0</slicepedia:table>
		<slicepedia:readingDifficulty>41</slicepedia:readingDifficulty>
	</slicepedia:metadata>
	<slicepedia:sliceContent>
		<slicepedia:div>The main Syrian armed rebel group has already refus
		ed to co-operate</slicepedia:div>
		<slicepedia:div>Gen Salim Idriss of the Free Syrian Army said he ca
		tegorically rejected the plan, and insisted that the most important
		thing was to punish the perpetrators of
			<slicepedia:keywordSearch search="chemical attack" >chemical
			attacks</slicepedia:keywordSearch>.
		</slicepedia:div>
		<slicepedia:dbpedia concept="http://dbpedia.org/resource/united_nat
		ions" relevance="8.4" style="background-color:FFCC99;">
			<slicepedia:div>If the talks in Geneva
				<slicepedia:verbChunk lemma="be" tense="simpre">are</
				slicepedia:verbChunk>
				successful, the US
				<slicepedia:verbChunk lemma="hope" tense="simpre">hop
				es</slicepedia:verbChunk>
				the disarmament process will be agreed in a UN Securi
				ty Council resolution.
			</slicepedia:div>
			<slicepedia:div> However, Russia
				<slicepedia:verbChunk lemma="regard" tense="simpre">r
				egards</slicepedia:verbChunk>
				as unacceptable any resolution backed by military for
				ce, or a resolution that
				<slicepedia:verbChunk lemma="blame" tense="simpre">bl
				ames</slicepedia:verbChunk>
				the Syrian government for
				<slicepedia:keywordSearch search="chemical attack" >c
				hemical attacks</slicepedia:keywordSearch>.
			</slicepedia:div>
			<slicepedia:div>Moscow has already objected to a draft resol
			ution that would be enforced by Chapter VII of the UN charte
			r, which would in effect sanction the use of force if Syria
			failed in its obligations.</slicepedia:div>
		</slicepedia:dbpedia>
		<slicepedia:div>Russia, supported by China, has blocked three previ
		ous draft resolutions condemning the Assad government.</slicepedia:
		div>
		<slicepedia:div>More than 100,000 people have died since the
		uprising against President Assad began in 2011.</slicepedia:div>
	</slicepedia:sliceContent>
</slicepedia:slice>
```

Figure 4.9: Slice Output Example[35]

4.4 Third Party Collaboration Procedure

Although any slicing collaboration between institutions is initially considered out of scope of this research (section 3.3.2), DR 5 requires the *generation of slices* and *underlying resources* needed to produce slices, to be *open* and support *possible* collaboration within third party institutions. Slicepedia should therefore be implemented in a manner which would easily enable such collaboration to occur.

To achieve this purpose TDR 7 additionally specified that any collaboration carried out, should heavily rely upon RDF technology. Furthermore, in order to support the access and browsing of RDF slicing data, TDR 6.a requires Slicepedia to provide search and delivery of slicing data through SPARQL queries. For these reasons, Slicepedia provides public access to the Virtuoso SPARQL endpoint (section 4.5), which enables any third party institution to openly browse slicing data produced by Slicepedia as open linked data. Consequently, this enables any third party institutions to collaborate by producing either i) additional slicing data if necessary or ii) slices for slice consumers of their choice. This section will therefore present how third party institutions would do so.

Section 4.2.3 mentioned how the task performed by the semantic analyser module within the Slicepedia pipeline was optional. This is because, as mentioned above, third party institutions can collaborate and contribute additional semantic meta-data to the Virtuoso repository by using their own set of preferred NLP algorithms. A slice consumer could hence request slices which were only annotated by independent institutions.

Although the sizing of fragments generated by the structural fragmenter module can be determined in real-time (section 5.2.3) based on the granularity of slices requested, if collaboration is to be supported, an a-priori decision with respect to the sizing of foundation fragments produced must be taken. This decision is required in order to enable various annotating institutions to refer to the exact same foundation fragment when assigning annotations. In such circumstances, and as in this prototype, the sizing of slices can be performed instead by the slice creator module of the slicer as

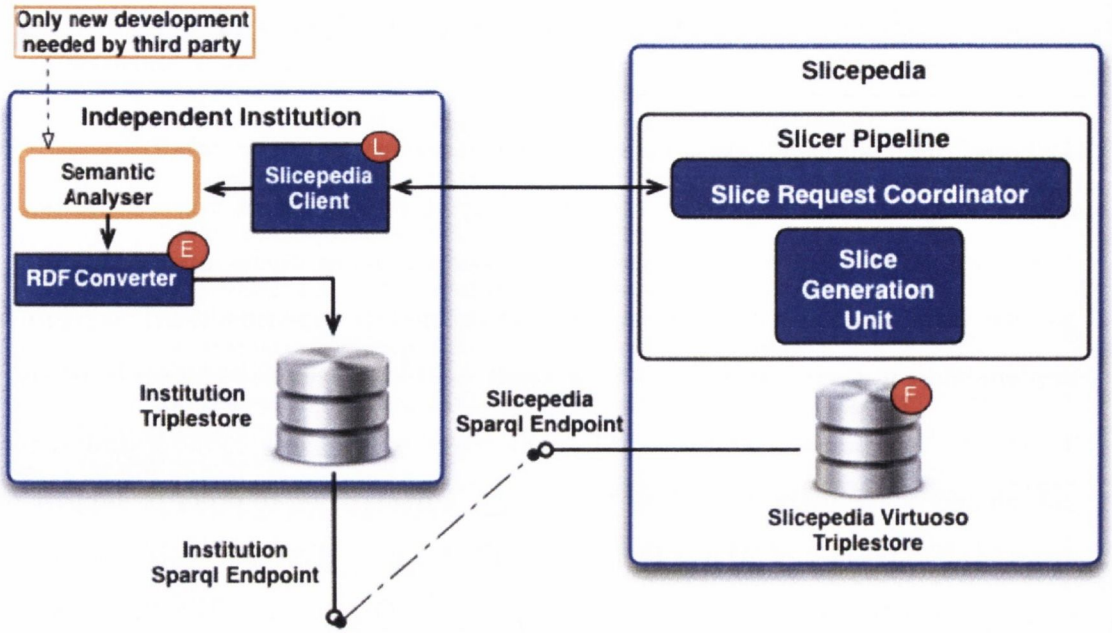


Figure 4.10: Third Party Collaboration

opposed to the structural fragmenter module.

As described in figure 4.10, an independent institution wishing to collaborate would simply act as a slice consumer to Slicepedia. Institutions would essentially implement their own version of the semantic analyser module presented earlier (4.2.3) and reuse the exact same *RDF converter* and *slicepedia client* module presented in more details in section 4.3.3.4.3 and 4.5 respectively.

The collaborating institution could request sets of slices from Slicepedia, similar to the *semantic analyser* module. In contrast with Slicepedia however, there is no requirement for slices requested by such institution to be naked. The proportion of meta-data to be included with slices requested would be based upon the dependencies of the chosen NLP algorithms (see section 2.4.4). Once the institution receives the requested slices, the NLP pipeline (implemented as part of its semantic analyser module) is run and meta-data produced is converted into RDF using the *RDF converter module*, stored in a triplestore and published as linked data.

In order to be reused, the institutions meta-data (published as linked data) must at

least refer to the public RDF URI pointing to the Slicepedia fragments annotated. The triplestore used to store meta-data produced by this institution can belong to the institution itself (if it supports this capability) or could be stored directly within Slicepedia itself (based on prior contractual agreements). In the latter case, in order to keep meta-data sources separate, meta-data originating from this independent institution would be stored within Slicepedia using a contextual graph [Carroll2005] dedicated to that specific institution. Doing so enables slice consumers to request slices, produced using meta-data originating from selected institutions, simply by specifying a list of contextual RDF graphs within the SliceQuery object (see section 4.3.4.3).

The end result of this collaboration is described in figure 4.11 consists of a *web of slices* constructed with third party annotations pointing to both annotations and fragments produced by Slicepedia, in turn pointing to original native web pages. Once the output of third party meta-data collaboration is accessible as linked data (DR 5), the Slicepedia slice generator unit can therein access this additional set of meta-data and hence also expand its CAS.

Since the underlying slicing data required to produce slices is available openly as linked data, slice generation performed by independent institutions could also be achieved in the event the Slicepedia slice generation process (described in the section 4.2.3) didn't meet all the needs of slice consumers or third party annotators. This scenario would lead to a ***partly-federated*** slicing architecture. In the event such a need arises, an independent slice generation unit would access Slicepedia's slice data using Virtuoso's SPARQL endpoint and proceed similarly as described above. In this context, Slicepedia would take the role of a hub, producing fragments of arbitrary size along with structural meta-data, which would subsequently be used as a basis for semantic annotation and slice generation collaboration between institutions.

Nevertheless, since the fundamental exchange and publication of slicing data relies upon the RDF protocol, there is no reason why only post-fragmentation modules of the slicing pipeline should be federated. A ***fully-federated slicing*** process therefore would involve federating not only the semantic analysis and slice generation

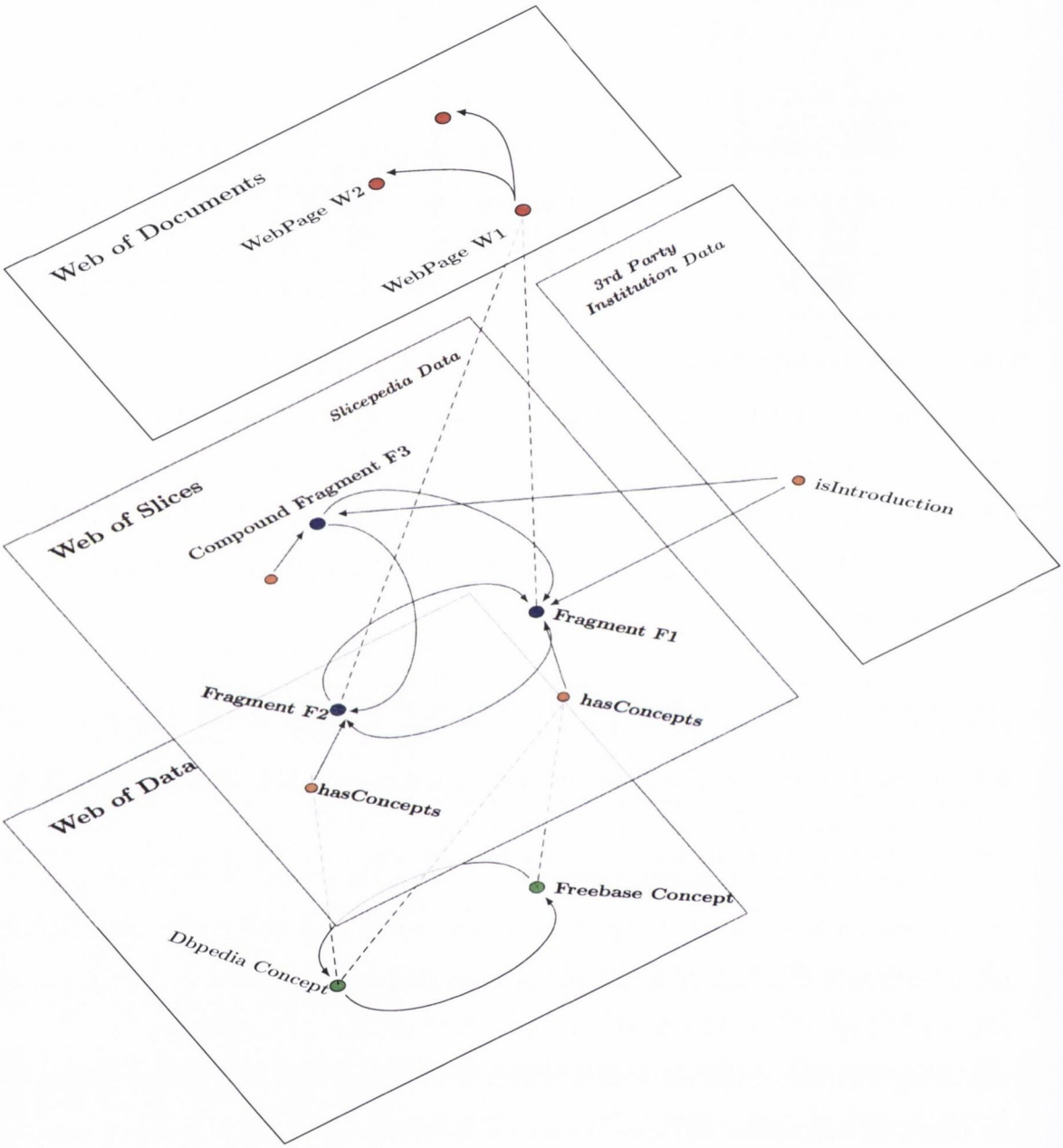


Figure 4.11: Web Of Slices Representation

modules but also the harvesting and fragmentation modules. Within this scenario, the set of fragments originally used to support annotation collaboration would now be distributed, with fragments being produced by various slicers and stored in various RDF repositories. This entails, therefore that individual native resources would possess various fragmented representations depending upon what slicers produced them, which could lead to important complexity (section 6.3). Another possible fully-federated slicing scenario could also involve original author to explicitly define various fragments of pages published on the web. In this scenario, each native web resource would act as a hub for any federated slicing process involving the reuse of itself.

4.5 Slicepedia Access

According to TDR 6, the slicer prototype implemented for the purpose of this research should provide multiple slice search & delivery alternatives. For this reason, the Slicepedia pipeline presented in section 4.2 was incorporated as part of an Apache Tomcat[36] web-server as depicted in figure 4.12. The pipeline was supplemented by a REST api interface (K) (section TDR 6) using the Jersey client web-service library [37].

In order to facilitate RESTFul communication between Slicepedia and slice consumers, a *SlicepediaClient module* (L) was also implemented. This module, available as a Jar library and reusable in any Java-based AHS, provides an object-oriented Slicepedia abstraction to slice consumers over RESTful communications. Hence, slice consumers are only required to create an object instance of Slicepedia, in which they pass a SliceQuery object. The SlicepediaClient module then takes care of all necessary RESTful communications. It converts the SliceQuery object into json (using the Gson library[38]) and forwards it to the Slicepedia Requests Coordinator (SRC) module, which converts it back into Java and proceeds as described in section 4.2.2.

As specified in TDR 6.c, cloud based applications should also be able to interact with Slicepedia as slice consumers. For this reason, a *Google App Engine (GAE) client module* (M) was also implemented to supplement the core slicer pipeline. This module behaves as a daemon within the Tomcat Web Server, and periodically checks a GAE pull task queue[39] (N) for any available SliceQuery objects. A pull task queue enables GAE applications to delegate specific tasks to external web-services[40]. Any SliceQuery object received from this pull task is then forwarded to the SRC module,

[36] http://tomcat.apache.org/

[37] http://jersey.java.net/

[38] https://code.google.com/p/google-gson/

[39] https://developers.google.com/appengine/docs/python/taskqueue/overview-pull

[40] The tasks needed to be delegated are stored in a queue and periodically checked by authorised external web-services. When these services detect new tasks to accomplish, these tasks are pulled out of the queue temporarily and leased for an arbitrary amount of time. When the external web-service accomplishes the task, the latter is deleted and any response object returned to the GAE application.

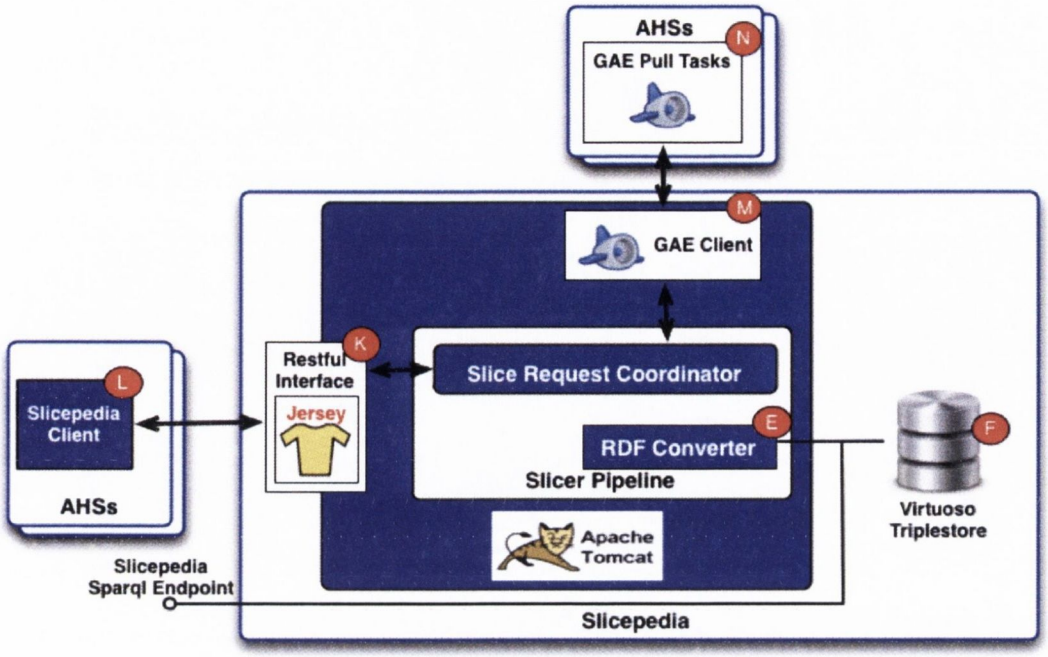


Figure 4.12: Slicepedia Interfaces

which subsequently returns slices requested.

TDR 4.a, requires Slicepedia to support ***keyword and conceptual searches***. To address this requirement, the slicer prototype, implemented for this research, makes it possible for slice consumers to specify a list of keywords and/or DBpedia URIs as part of a SliceQuery object. As is described in section 4.3.3.1, the Virtuoso triplestore (F) supports SPARQL full text searches of RDF object values. In other words, it is possible to declare that specific resources within triples (with a particular given predicate or graph) get indexed. For this reason, each fragment produced by Slicepedia's structural fragmenter, stored and assigned a given predicate, is indexed by the Virtuoso triplestore. Whenever, a SliceQuery object containing keywords is forwarded to the slice generator unit, the latter simply converts this object into a SPARQL query (as described in section 4.2.3) according to Virtuosos SPARQL full text searches specifications.

A similar process is undergone with respect to conceptual searches. As described above, these are carried out (from the point of view of slice consumers) by providing a

list of URIs referring to specific DBpedia RDF concepts. Since Slicepedia's semantic analyser uses, as part of its NLP pipeline, a concept tagger which assigns DBpedia concepts to fragments produced by the structural fragmenter (see section 4.2.2), performing a conceptual search therefore simply requires the SliceQuery object converted into SPARQL to possess a SPARQL statement only matching fragments assigned with a predicate object value equal to one or more URIs specified by slice consumers. Fragments identified both through a conceptual or keyword search are then subsequently merged into slices, by the slice generator module, depending upon granularity requirements. Since both keyword and conceptual searches are ultimately converted into SPARQL queries, both searches can easily be performed simultaneously by producing queries containing both SPARQL statements.

Finally, as required by TDR 6.a and as mentioned in section 4.4, Slicepedia provides a public SPARQL endpoint simply consisting in the Virtuoso's SPARQL endpoint. This enables any third party institution to have direct access to slicing data produced by Slicepedia.

4.6 Slicepedia CAS

The resulting CAS offered by Slicepedia (see table 4.4) consists of three right-fitting dimensions (Content Style, Granularity and Annotation Type) including ten adaptation variables that can be arbitrarily combined to suit various content requirements over any relevant open corpus resources identified. An example of a slice request hence could consist of the following: *slices should originate from only a specified list of urls and have a granularity ranging from 3 sentences up to 3 paragraphs. They should cover the topics of whale migration, atlantic ocean or hunting and should have a Flesh reading score ranging from 45 to 80. They should not contain any tables or bullet points lists but be focused on the specified topics (ie: exclude content not on these topics). Slices should contain between 7 and 15 annotations consisting of verbs conjugated at the past perfect continuous and should be delivered in xml format.*

<table>
<tr><th colspan="3">Slicepedia CAS</th></tr>
<tr><td colspan="3">Content Style</td></tr>
<tr><td colspan="2">Format</td><td rowspan="2">Reading Difficulty</td></tr>
<tr><td>Bullet Points</td><td>Table, Prose, Code, Form</td></tr>
<tr><td colspan="3">Granularity</td></tr>
<tr><td>Sizing</td><td colspan="2">Content Focus</td></tr>
<tr><td>Para/Sentences/Wrds</td><td>Concept</td><td>Annotation</td></tr>
<tr><td colspan="3">Annotation Type</td></tr>
<tr><td>Verb Chunk</td><td>DBpedia Concept</td><td>tag replacement/removal, original sources, nbr of annotations</td></tr>
<tr><td colspan="3">Keyword Search</td></tr>
</table>

Table 4.4: Slicepedia CAS

4.7 Conclusion

This chapter presented a detailed description of the overall implementation carried out for the purpose of this research. The resulting prototype was implemented based upon the design requirements enunciated in the previous chapter, with the intention of contributing to objective two of this thesis. Individual components, as well their overall interactions, were described in detail. The following chapter presents an evaluation of this prototype, both with respect to individual components as well as the overall system.

Chapter 5

Evaluation

5.1 Introduction

The first chapter of this thesis presented the overall research question investigated within this document from which three main objectives were derived. The first objective was addressed in chapter 2 by investigating the state-of-the-art of digital content analysis and AHSs content supply techniques, while chapter 3 and 4 contributed to the second objective through the design and implementation of a novel open-corpus content slicing service named Slicepedia. This chapter aims at addressing the third objective of this thesis which aims to *evaluate the extent to which such a service can improve the reuse of open corpus resources within independent AHSs.* A detailed description of the evaluation performed for this purpose, is therefore presented in this chapter.

As mentioned in section 1.4, the overall research process chosen for this investigation involved focusing as a priority upon the overall end-to-end reuse of open corpus resources (from harvesting to reuse within independent AHSs) as opposed to individual steps performed as part of this reuse. Each evaluation presented in this chapter was hence designed taking into account the intention to validate, as a first step, overall content/AHS agnostic approach presented by this thesis as opposed to achieving the best possible performance at each step involved in the reuse. Whenever individual steps are given more attention, this is to ensure that this does

not compromise the resulting end-to-end approach overall.

In order to evaluate how well this research objective was achieved, the evaluation was divided into four parts. The first evaluation presented in this chapter consists of a detailed analysis of the structural content fragmentation approach presented in section 2.4.3. This novel approach to content fragmentation was selected for the prototype implementation of this research (see section 4.3.3.4). However, as presented in the second chapter of this thesis, the ability to fragment content in an agnostic manner is not currently fully established and involves many compromises. This component represents a decisive step of any content slicing service since a bad performance would affect all subsequent pipeline components (and could therefore potentially undermine the overall end to end content/AHS agnostic reuse approach investigated as part of this thesis). For this reason an in depth analysis evaluating the performance of this fragmentation approach is presented. The analysis was carried out with respect to a range of technical requirements (specified in section 3.3.3), (unavailable at the time the slicer prototype was being implemented), was necessary. The analysis focused upon the performance of several variations of the algorithm with respect to content dependency, multilingual performance, sizing and time performance among others.

Following this analysis, section 5.3 presents an examination of how each Slicepedia design requirement presented in chapter 3 was addressed in the implementation carried out for the purpose of this research.

Section 5.4 subsequently presents an evaluation of the overall performance of the slicer. This was achieved as part of a user-trial experiment involving real-life user needs. Given a determined range of diverse and undetermined source of open corpus resources, the evaluation focused upon assessing content slicing as an approach in general to open corpus reuse improvement, as well as a examining more specifically the performance of a content/AHS agnostic approach to slicing.

Finally, section 5.5 presents an experiment that investigates the ability of a content/AHS agnostic slicer to supply recomposable resources to independent AHSs, with content reuse scenarios unknown in advance to slicer designers. This experiment

was performed as a trial involving computer science university students within an authentic educational scenario. The rest of this chapter hence provides details on each evaluation presented above. Each section presents the experiment objective, the evaluation metrics employed, the experiment methodology used and an analysis of the results observed.

5.2 Densitometric Fragmentation Evaluation

5.2.1 Experiment Objectives and Hypothesis

Among the set of design requirements enunciated in chapter 3, TDR 3 presented a list of characteristics which a fragmentation algorithm, used within a slicer, should possess (section 3.3.3). This list is repeated below in table 5.1 for the readers convenience.

Among the wide variety of algorithms designed for the purpose of content fragmentation, DCF [Kohlschutter2008a] was identified in section 4.3.3.4.1 as the most promising approach reviewed with respect to these fragmentation requirements. Nevertheless, although the authors of the approach claimed to achieve high accuracy and time performance, at the time the slicer was being implemented for the purpose of this research, it appeared to the authors knowledge that no independent evaluation of the approach had been carried out. Moreover, performance of this approach with respect to a range of characteristics of importance to slicers (such as multilingual support, sizing control etc.) were unavailable at the time of writing. As mentioned within the previous section, this thesis focuses upon the overall end-to-end reuse

ID		Technical Design Requirements
3		Fragmenter characteristics:
	a	high speed and fully-automated processing of large volumes of pages
	b	domain and language independent
	c	content type agnostic
	d	page-level fragmenter
	e	tag meaning independent
	f	tag structure independent
	g	predictable fragment sizing control

Table 5.1: Extract from Technical Design Requirements

of open corpus resources through a content slicing approach. Nevertheless, since fragmentation represents a decisive step within the context of a content slicing system, a low performance of this component could significantly affect the results obtained with respect to the overall reuse of resources through slicing. For this reason, this section deviates temporarily from the overall objective of this thesis and presents a full evaluation of DCF algorithms aimed at identifying the strengths and weaknesses of this approach across a range of characteristics critical for content slicing, in order to determine its suitability for slicing purposes.

As was described in section 3.4.2 and 4.3.3.4.1 requirements TDR 3.d, TDR 3.e and TDR 3.f are fulfilled based upon the approach taken by DCF to process pages. Requirements TDR 3.a, TDR 3.b, TDR 3.c and TDR 3.g however require the support of a detailed analysis. The following section therefore presents the set of hypothesis considered for this evaluation.

The set of hypothesis tested during this analysis are presented below:

Hypothesis:

H1) DCF is a suitable fragmentation approach for incorporation within a slicer pipeline

H1.a) DCF provides predictable fragment sizing control (TDR 3.g)

H1.b) DCF achieves a high performance regardless of content types (TDR 3.c)

H1.c) DCF provides predictable parallel accuracies across multiple languages (TDR 3.b)

H1.d) DCF achieves time performances within an acceptable range[1] (TDR 3.a)

H1.e) DCF provides predictable and adjustable trade-offs across characteristics of relevance for slicing purposes

[1] hundreds of thousands of pages processed within a matter of hours using a standard computer (see section 3.3.3)

5.2.2 Experiment Set Up

In this evaluation, i) plain fusion, ii) smooth fusion, iii) pure rule-based and iv) smooth-rule-based fusion variations, described in section 3.4.2 were implemented and tested for the purpose of this analysis. These algorithms were evaluated over a parallel multilingual corpora of approximately 20,000 pages acquired from Centre for Next Generation Localisation (CNGL)'s commercial partner Microsoft[2]. This corpus consisted of MS Office[3] manuals in four different languages (English, French, German and Spanish).

5.2.2.1 Experiment Metrics

With the intention of keeping consistent with the literature in the field, and for comparative purposes, all accuracies measured within the subsequent evaluations were carried out using the standard ***Adjusted Random Index (ARI)*** metric [Strehl2003].

The Rand Index [Strehl2003] measures the similarities between two clustering methods by determining the number of agreements within two vectors. Within the context of this analysis, these vectors represent the fragment each particular tag belongs to. The ARI [Yeung2001] is simply a normalised extension of the Rand measure with agreements between clusters rated using values ranging from zero (no agreement) to one (perfect agreement).

[2]http://www.microsoft.com/

[3]http://office.microsoft.com/

Γ — Π	Π_1	Π_2	..	Π_p	**Sums**
Γ_1	δ_{11}	δ_{12}	δ_{13}	δ_{1p}	b_1
Γ_2	δ_{21}	δ_{22}	δ_{23}	δ_{2p}	b_2
..	..	..	..	..	..
Γ_g	δ_{g1}	δ_{g2}	δ_{g3}	δ_{gp}	b_g
Sums	a_1	a_2	..	a_p	

Table 5.2: Adjusted Random Index Contingency Table

Hence, given two final block arrays Γ and Π obtained after being processed through a DCF algorithm, each containing a set β of n atomic blocks, distributed within two sets of final compounded blocks (or fragments), namely $\Gamma = \Gamma_1, \Gamma_2, \ldots, \Gamma_g and \Pi = \Pi_1, \Pi_2, \ldots, \Pi_p$, a contingency table (table 5.2) can be created which holds δ_{gp} the number of atomic blocks held in common between each final compounded block. The ARI derived from this contingency table can thereafter be computed according to equation 5.1.

$$ARI = \frac{\sum_{gp} \binom{\delta_{gp}}{2} - \left[\sum_g \binom{a_g}{2} \sum_p \binom{b_p}{2}\right] / \binom{\delta}{2}}{\frac{1}{2}\left[\sum_g \binom{a_g}{2} + \sum_p \binom{b_p}{2}\right] - \left[\sum_g \binom{a_g}{2} \sum_p \binom{b_p}{2}\right] / \binom{\delta}{2}} \tag{5.1}$$

A ***sizing index*** was also used in section 5.2.3 to measure the resulting fragment sizes of pages processed (expressed as a percentage value). For each page processed, this metric was computed by getting the difference in block numbers between pre and post fusion, and normalising each value according to the total number of blocks present prior to fragmentation. Hence values close to a hundred correspond to very large fragment sizes while those closer to zero represent small fragment size.

$$SizingIndex = \frac{\sum Blocks_{PreFusion} - \sum Blocks_{PostFusion}}{\sum Blocks_{PreFusion}} \tag{5.2}$$

5.2.2.2 Implementation Validation

So as to validate the DCF algorithm implementations used within this research, a direct comparison with existing state of the art systems was required. Although access to the original Kohlschtter [Kohlschutter2008a] DCF algorithm implementations was unavailable at the time this analysis was performed, sample pages/fragmentations data set used in the original publication were successfully acquired and used as a baseline. Sample pages were processed with line wrapping W_x and threshold parameters V_{max} respectively set to 80 and 0.38 (according to the experiment performed in the original publication). ARI similarities were computed between resulting fragments and baseline fragments.

As described in figure 5.1, plain fusion as well as smooth fusion algorithm implementations achieved 90% agreement with baseline fragmentations, while both rule-based variations achieved 85% accuracy. The high similarity score achieved by plain fusion is a very important result, in terms of confirming the integrity of the implementations carried out for this research, since the plain fusion algorithm is used as a foundation for all other fusion variations of DCF algorithms. Nevertheless, similarity scores of only 70% for smooth fusion is surprising.

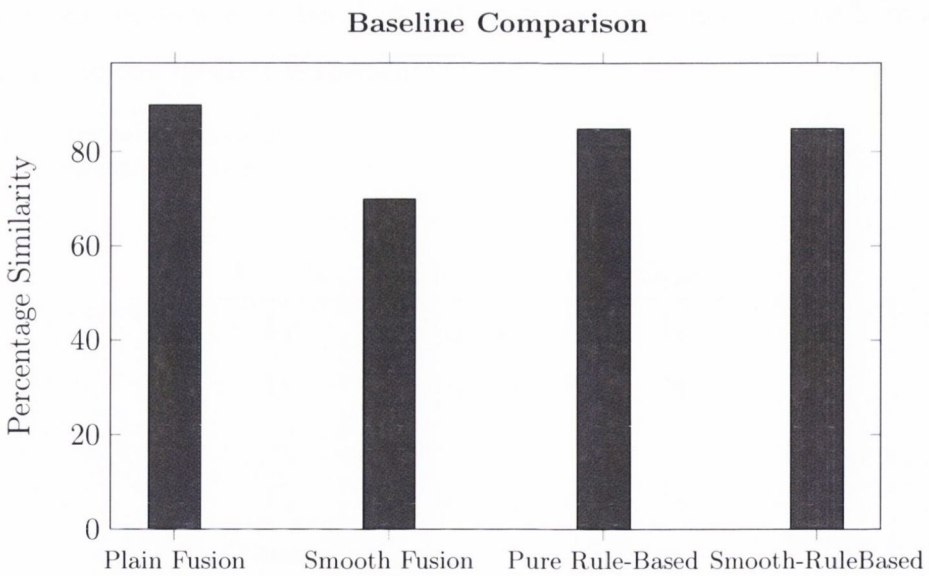


Figure 5.1: Baseline Comparison

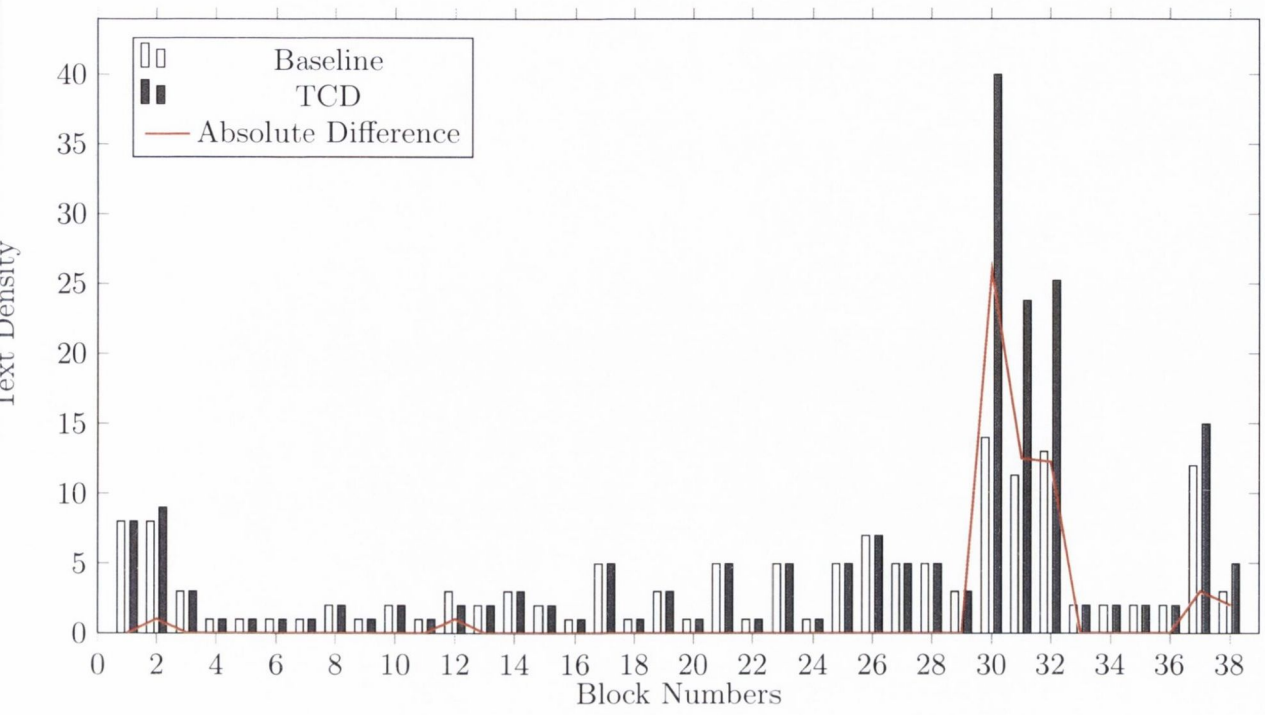


Figure 5.2: Pre-fusion Text Densities Comparison

Although the similarity scores are quite high, in order to investigate whether these differences were caused by the fusion algorithm implementation differences, a comparison between the initial text density measurements prior to any fusion was computed (figure 5.2). As can be seen in figure 5.2, an important difference in text density close to block thirty was discovered.

An initial observation of the total number of token parsed between the two implementations was initially thought to be the cause of these discrepancies. A comparison between text densities which both implementations should theoretically depict (based on their respective number of tokens taken into account), with those produced by both implementations was therefore plotted (figure 5.3). Although the text density values produced by the implementation carried out for the purpose of this analysis matches perfectly theoretical text density values, figure 5.3 shows how the baseline implementation diverges significantly.

A deeper analysis of the baseline results, with respect to the number of tokens within each block and text density values produced this implementation was hence

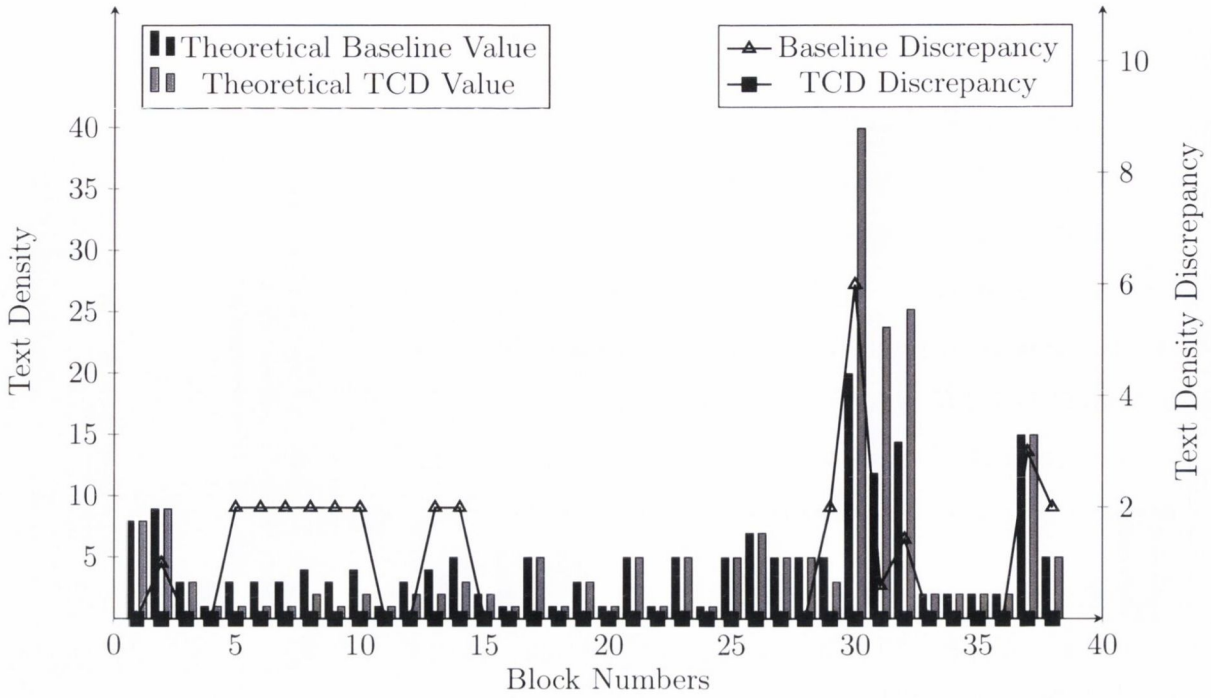


Figure 5.3: Baseline and TCD Implementations Discrepancies

performed. Figure 5.4 presents two different regions of discrepancies in the baseline data set. The first half of the block array (top graph) represents a region where block line numbers are equal to one, while the region in the second half (bottom graph) depicts blocks with higher line numbers. For the first case, according to the theoretical description of the algorithm (section 3.4.2), whenever a block line number is equal to one, the text densities should equal the number of tokens located in such a block. As can be seen in the top graph, a major deviation of this rule occurs in particular between blocks five to fourteen, which explains the error depicted in figure 5.3. In the second case however, since the baseline implemented block text density calculation was unavailable, it was impossible to determine the cause leading to these errors. Nevertheless, as the bottom graph points out, significant variations occur close to block thirty with errors representing nearly half of the measured text density.

Conclusion: The initial verification of DCF algorithm's implementation, carried out for the purpose of this research, depicted very high correlations with the available baseline data set. Differences measured and investigated, revealed a divergence between baseline calculations and expected theoretical values. Densitometric

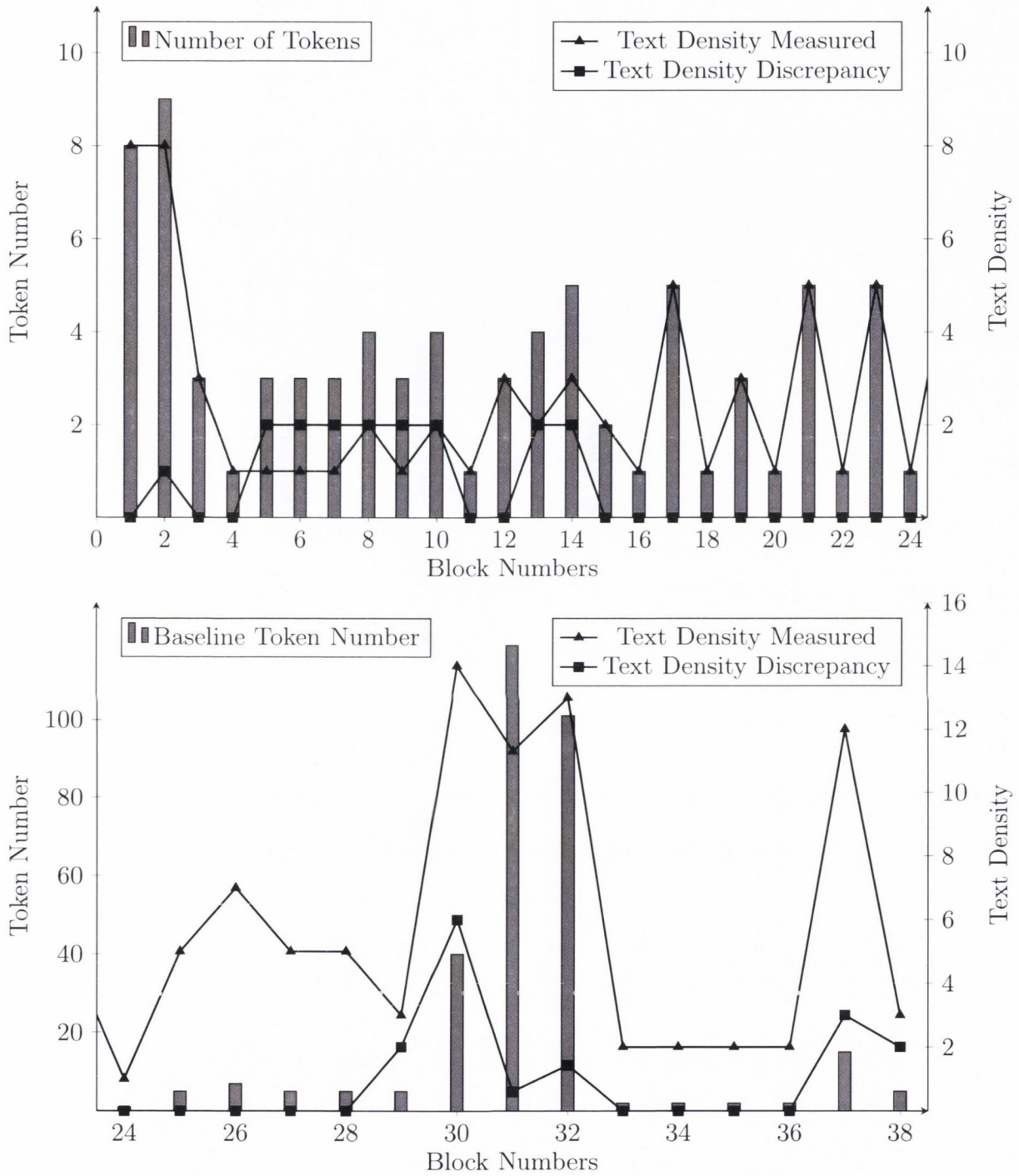


Figure 5.4: Baseline Text Density Theoretical Discrepancies

calculations performed by the implementation carried out for this research, on the other hand, matched these theoretical values perfectly. For these reason, this implementation can be confidently stated as being validated as well as representing a closer match to the original theoretical representation of the algorithm originally presented by Kohlshutter [Kohlschutter2008a].

5.2.3 Densitometric Content Fragmentation Sizing

TDR 3.g , requires the fragmentation approach to provide control over the resulting size of fragments produced. Consequently, the ability to predict and control the size of fragments produced by DCF algorithms was measured (H1.a) with respect to threshold V_{max} and word wrapping W_x parameters.

Figure 5.5 depicts the results obtained for DCF, using plain fusion, executed over the English subset of the corpus (approximately 5000 pages). As one can see, the graph depicts a linear increase in size of fragment output across a wide range of values ranging from ten to ninety. Both threshold V_{max} and word wrapping parameters W_x are clearly directly correlated with this property. Although V_{max} values have a determinant impact upon fragment size, for a given V_{max} value, the gradient of size change is determined by W_x values. In other words, as W_x decreases, a slight increase in threshold value V_{max} will result in large fragment size change. As will be demonstrated in the following sections, this characteristic is observed repeatedly for

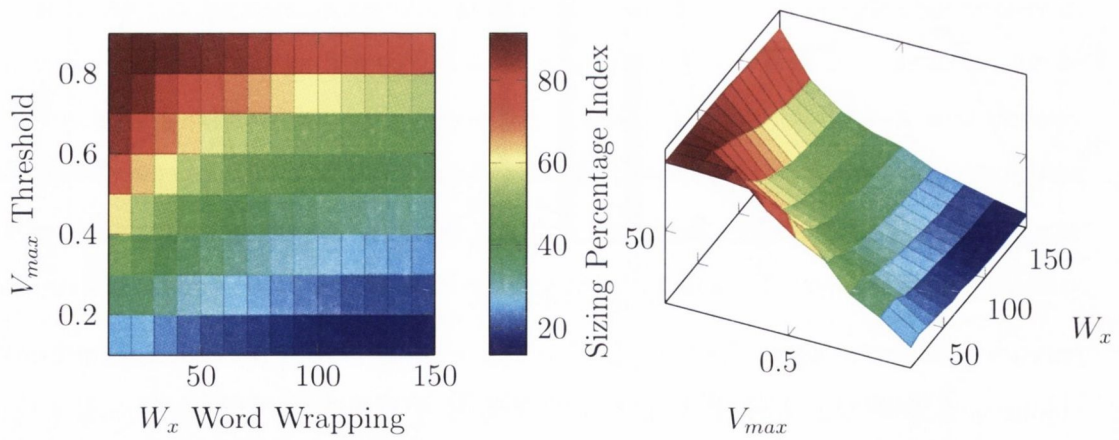


Figure 5.5: Plain Fusion Sizing Analysis

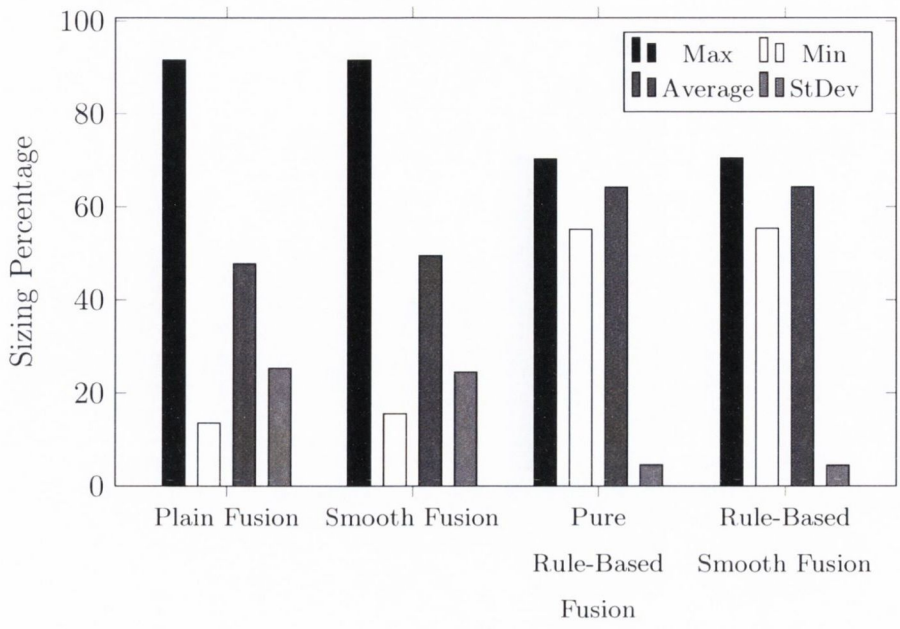


Figure 5.6: Densitometric Fragmentation Sizing

various properties measured. Hence this observation contradicts the notion stated by Kohlshutter, which suggests the word wrapping value W_x should be fixed arbitrarily. This analysis presents a mutual dependency between two parameters V_{max} and W_x, which must be taken into account when selecting adequate V_{max} values.

Moreover, as shown in figure 5.6, a clear distinction between non rule-based and rule-based fusions can be observed. Non rule-based variations provide a wide range of possible sizes with a difference of 76% between smallest and largest fragments produced. Rule-based variation on the other hand, only provide a difference of 15%. These values suggest non-rule-based fusions provide a much larger sizing range with respect to their rule-based counter part. However, while rule-based fusions provide a stable standard deviation with respect to V_{max} and W_x parameters, non rule-based fusions possess a standard deviation of up to 30% for V_{max} values ranging between 0.4 and 0.7, which would suggest lower predictability of fragment output sizes.

Conclusion: The results obtained in this analysis show that DCF does provide the ability to *control* the size of fragments produced which validates (H1.a). However these results also suggests the ability to accurately *predict* the output size of fragments involves a trade off between the range of fragment sizes desired and the predictability

of fragment sizing. A rule-based fusion DCF approach will offer a small range of possible fragment sizes, however most fragments produced will conform to the requested size. Non rule-based DCF on the other hand provides a much wider range of possible fragment sizes however the size of some fragments produced will tend to be more variable than its rule-based counter part. Finally, as mentioned earlier, the assumption that W_x values should be arbitrarily set appears to be erroneous. For this reason, all subsequent analysis will take into account results obtained with respect to V_{max} and W_x parameters.

5.2.4 Fragmentation Accuracy

5.2.4.1 General Accuracy Performance

The next experiment consisted in analysing the fragmentation accuracy of each DCF algorithm implemented over the English subset of the corpus (H1.b). Since manually annotating a corpus of this size was unrealistic (section 5.2.4.2) the next best alternative to human annotation consisted in using a rule-based system to semi-automatically annotate this corpus [Kohlschutter2008a].

An exhaustive search for tags contained in these files listed sixty-seven individual tag types from which twenty were used as rule triggers. As a matter of fact, these tags resulted in providing valuable consistent cues in revealing the structural composition of these manuals. An inspection of these rule-based annotations in comparison with randomly sampled human annotated pages obtained an average ARI of 95% accuracy, which was considered adequate for this experiment.

The result of fragmenting the English corpus using DCF algorithms is presented in figure 5.7. This graph depicts the ARI accuracies obtained by DCF plain and smooth fusion variations with respect to V_{max}. The expected densitometric characteristic bell curve is observed and a peak is obtained at approximately $V_{max} \approx 0.4$ and $W_x \approx 80$, which confirms Kohlschutter observations. The gradient effect of W_x discussed in the previous section was also noticeable upon the accuracy.

However, as one can see the results are very disappointing. Although accuracies

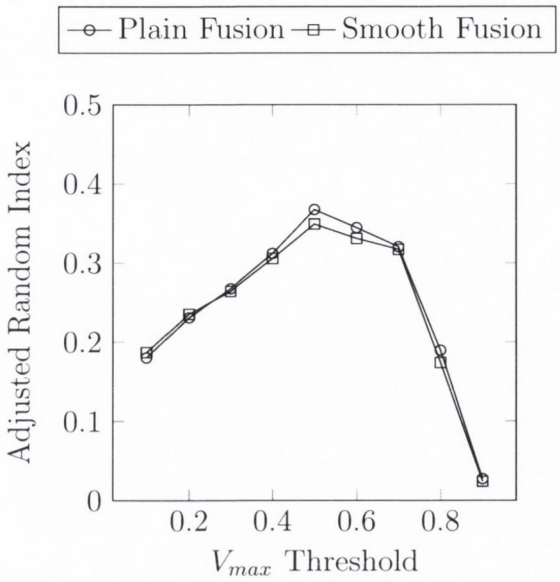


Figure 5.7: General Accuracy Results

reaching 57% for plain fusion were expected [Kohlschutter2008a], this analysis measured a maximum mean accuracy of only 37%, which is 20% lower than anticipated. Moreover, the DCF smooth fusion variation showed only a minor improvement.

However, taking a closer look at the data collected, a substantially large standard deviation was measured, varying between 21% and 31% for all algorithm implementations. Hence although the average accuracy is quite low, some pages do reach accuracies of up to 68%, which is closer to the range originally expected. The next experiment thus aimed at investigating further this matter by examining any possible DCF algorithms content type dependency, which would explain these disappointing results.

5.2.4.2 Content Type Accuracy Performance

Based on the unsatisfactory general accuracy results obtained in the previous section, an experiment investigating the wide standard deviation in accuracies measured, was performed. The objective was to investigate any possible content type dependencies of DCF algorithms. Such a dependency is critical within the context of a slicing system since the aim is to process a vast variety of pages regardless of their original

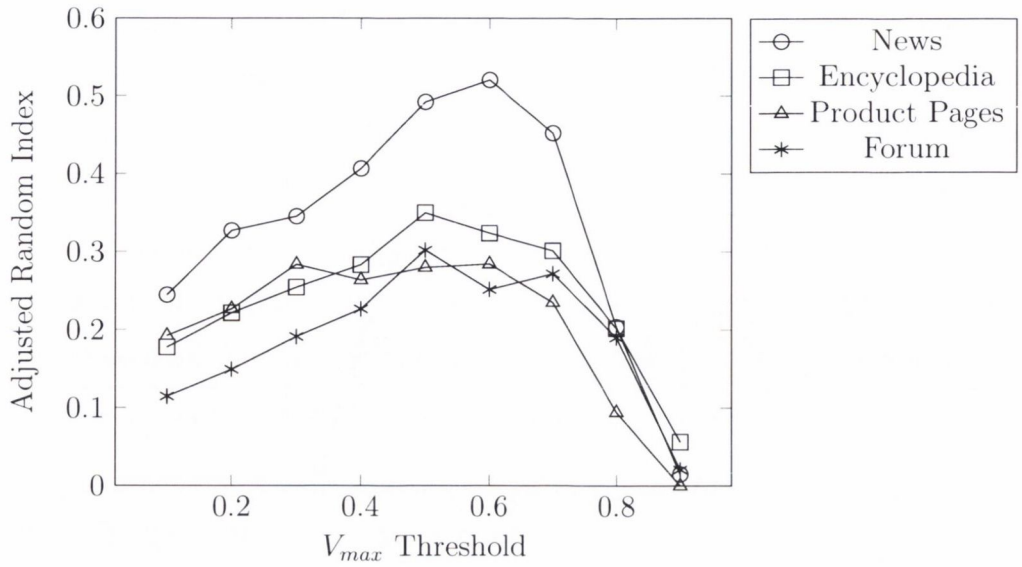


Figure 5.8: Plain Fusion Content Type Accuracy Results

usage (section 3.3.3) .

As mentioned previously, content types within this experiment refer to four general page types namely, Encyclopedias (e.g. Wikipedia.org), Forum, Product (e.g. Amazon.com) and News pages. Five human annotators manually annotated a sample corpus of 250 pages from each content type with an average annotating time of 4min/per pages. In order to provide a comparison baseline with other studies, the news set consisted of a subset of the corpus used in the original paper [Kohlschutter2008a]. Pages for other content types were randomly selected among standard websites representing each category.

Figure 5.8 presents the results obtained using plain fusion DCF across content types. Very similar results were observed for the smooth fusion variation. As can be seen, the results strongly present differing accuracies with respect to the content type being fragmented. News and encyclopaedia pages achieve accuracy values close to fifty-five percent, which matches expected accuracies claimed by Kohlschutter. However product and in particular forum pages depict very poor accuracies in the low twenties.

A closer look at a typical forum page provides an insight as to why such a difference

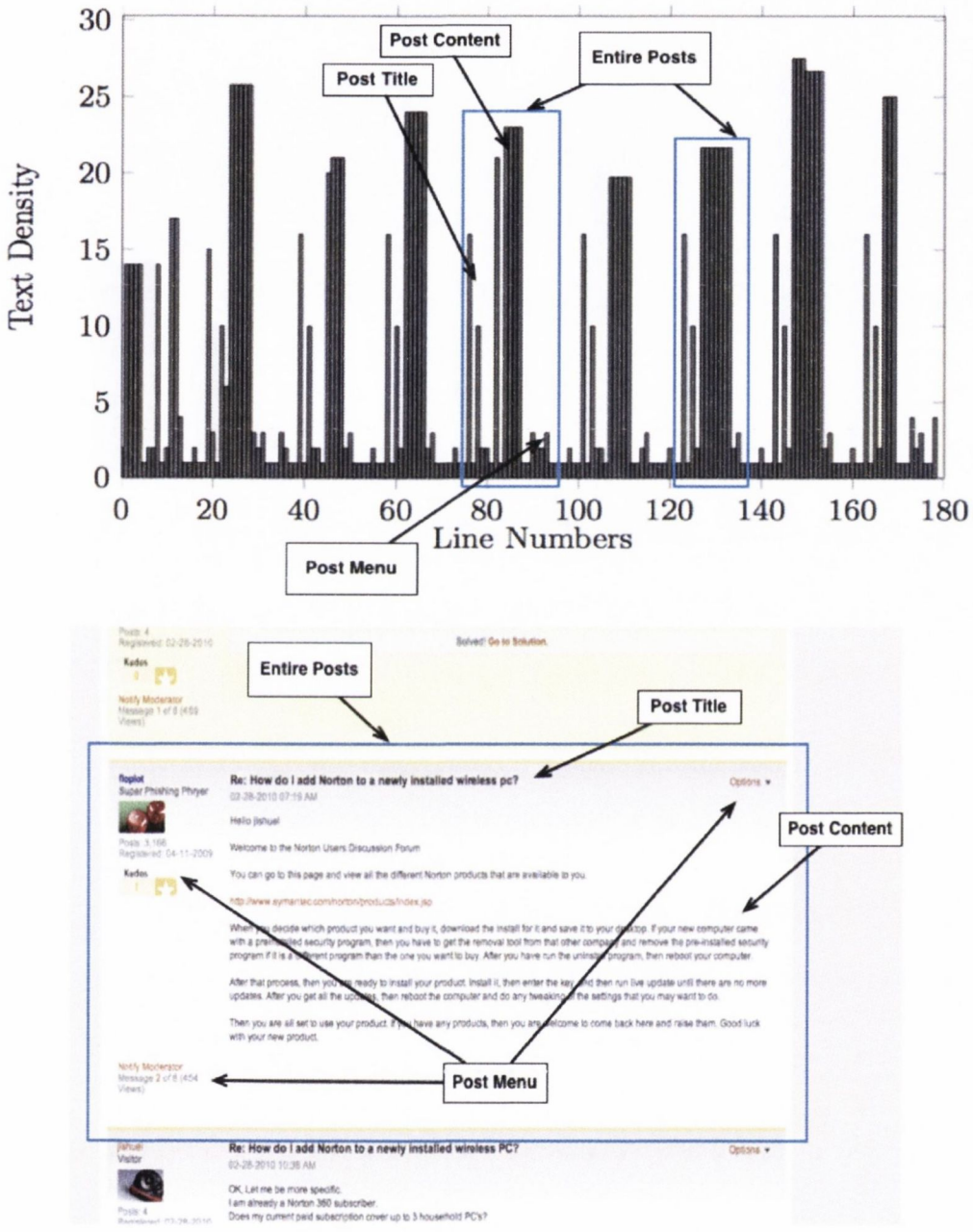


Figure 5.9: Forum Pages

in fragmentation accuracy could occur. As figure 5.9 points out, each forum post contains both the actual content of the post along with post menus (product page densitometric values depict the same pattern). As the text density diagram illustrates, these menus have a very low text density while the title and content depict high density. A post therefore contains elements with very high and very low density, which DCF algorithms are designed to separate from each other. This results in the title and menus of each post being separated from the post content, which could explain the poor performance of forum content over all fusion versions.

Conclusion: Hence, this analysis strongly suggests a DCF algorithm dependency upon the type of content being fragmented. Higher accuracies are obtained for pages containing fragments with continuous regions of similar high or low densities (such as News and Encyclopedia pages) in opposition to fragments with alternating high/low densities (Forum, Product pages). Nevertheless, figure 5.8 does appear to suggest DCF performances depict an intuitive ranking of content type accuracy by degree of editorial control. A lot of care must hence be taken prior to selecting this algorithm with respect to the type of content envisaged to process. As TDR 3.c requires the ideal content fragmenter, to be used within a Slicer, to accurately process pages regardless of their content type, this discovery highlights an important limitation for the purpose of slicer implementation and hence fails to validate hypothesis H1.b. If DCF is to be chosen as the selected approach for a slicer, improvements to the algorithm should be made to reduce this content type dependency (section 6.3), or the range of open-corpus material eligible to be targeted should be restrained temporarily. In the event further investigations fail to resolve this content type dependency, the use of several content fragmenter approaches (selected by classifying resources in content types), as proposed by [Weissig2009], could be perhaps envisaged. Alternatively page classification could also be used to change parameters heuristic according to the resource content type as performed by [Gupta2005b].

5.2.5 Multilingual Fragmentation

As pointed out in section 3.3.3, open-corpus resources available on the WWW are by nature multilingual. Although multilingual slicing is considered out-of-scope for the purpose of this research (section 3.3.3), TDR 2 requires the implementation of Slicepedia to either select components supporting the processing of multilingual content or alternatively to suggest some alternative component which can support additional languages.

Moreover, in addition to the fact that open-corpus resources are by nature multilingual, other possible use-cases involving parallel corpora in multiple languages (e.g. typical enterprise product manual localisation scenario), could make use of slicing systems. Hence, in addition to supporting the fragmentation of resources in multiple languages, the ability also to predict the similarity and accuracy of fragments produced from a set of multilingual parallel documents is important. The following analysis therefore measures the linguistic impact of resources processed through plain fusion DCF (H1.c).

In this experiment the full 20,000 pages of the multilingual (English, French, German, Spanish) parallel corpora were used. This corpus represents a good localisation use case example since it consists of Microsoft Office manuals translated in four languages. Within this corpus, each English file possesses an XML structurally identical twin. The only difference for each parallel file set is the linguistic content within each tag. Hence, any fragmentation variations between equivalent files could theoretically only be due to linguistic differences within the files. In reality however, minor differences were actually noticed between the XML parallel corpora documents. Nevertheless, following the conversion of these files into densitometric block arrays (section 3.4.2), a 100% block array similarity (excluding text densities of course) was measured between each combination. Hence although very minor differences did exist between the XML structure of each parallel file set, these were removed following a densitometric block conversion. Hence, prior to fusion, each individual page possesses the exact same number of densitometric blocks than its multilingual parallel counter part.

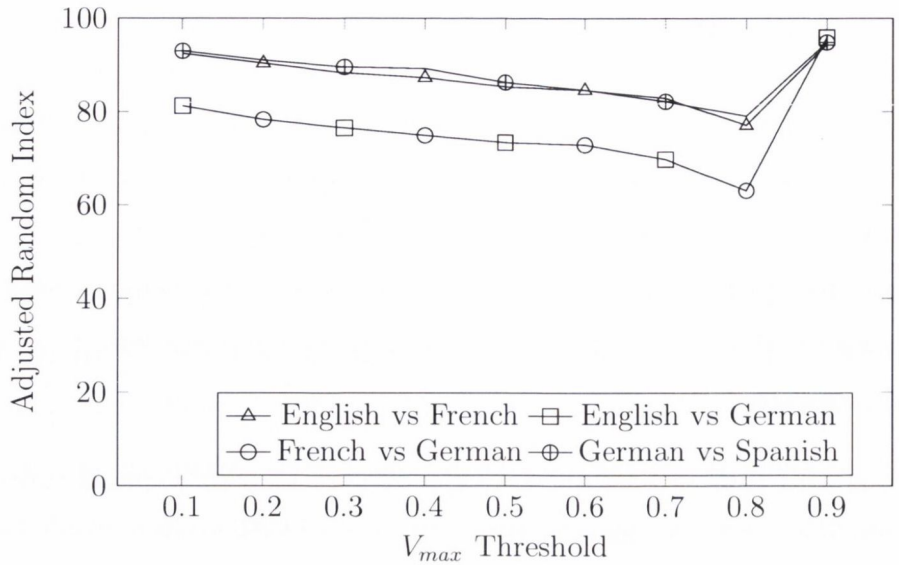


Figure 5.10: Multilingual Analysis

Figure 5.10, presents the results obtained by comparing each fragment language conbination for every parallel file set. Language combinations not presented within thegraph were omitted for clarity purposes as they achieved very similar results to theGerman w.r.t Spanish combination. Surprisingly, W_x had no noticeable impact upon the resulting fragment variations. However, V_{max} clearly determines the level of similarity expected between different language fragments. Fragment similarities, although very high, will decrease in average by 2% for every 0.1 increase in V_{max} ($\forall V_{max} < 0.8$). Past this value, the accuracy sensitivity increases very fast with respect to V_{max} ($\forall\ V_{max} > 0.8$) Furthermore, a standard deviation of only 2% was measured across all language combinations, which suggests a very high predictability in the fragment similarity expected for given parameters.

Finally, figure 5.10 shows how every language combination present very similar accuracies with respect to threshold values. This is true with the exception of French/German and English/German combinations which consistently portray accuracies 10% lower than others. Although these language combinations do achieve high predictable performances, this result suggests language combination decisions must be taken carefully. Hence, if DCF is to be selected for a slicer targeting multilingual open-corpus resource, additional experiments will need to be carried

out to investigate the underlying reasons for this difference, as well as identifying what linguistic combinations will achieve lower performances.

Conclusion: From this experiment it is apparent that, although a densitometric fusion approach to content fragmentation is linguistically agnostic (meaning it can fragment many languages[4]), a decrease in similarity across resulting language fragments, which is very predictable, must be taken into account while choosing optimal fragmentation parameters for parallel corpora fragmentation. It also suggests that difference linguistic combinations might achieve lower parallel precisions. Nevertheless, this analysis presents results which are very promising with respect to the ability of DCF to support multilingual fragmentation since accuracies depicted achieve a high score with high predictability. For this reason, this analysis considers this approach to fragmentation to be meeting requirement TDR 3.b and therefore also validates H1.c. Finally, since the multilingual parallel corpora available for this experiment only provided European languages, an experiment repeating and extending this analysis over languages with differing tokenization behaviour (such as Chinese for example) should also be performed in the future (section 6.3).

5.2.6 H1.d: Time Performance

As mentioned in section 3.3.3, the need to process pages at very high speed is critical to content slicing systems. For this reason, TDR 3.a requires the content fragmentation to be performed at high speed (H1.d). Ideally, an entire corpus should be fragmented on a standard machine within the order of hours (section 3.3.3). Hence each fragmentation should occur at speeds under 10ms in average per page. Although DCF algorithms can perform fragmentations regardless of specific tag structures, they do nevertheless rely on the presence of an existing structure. In other words, using such an approach to fragmentation over pages where all of the content is enclosed within one pair of tags, would clearly depict very high time performance. Within this context however, no fragmentation would occur since this approach considers leaf tags as initial atomic blocks (section 3.4.2). The aim of this

[4]Only European languages were taken into account for this experiment

analysis therefore was to determine the impact of fragmentation parameters upon processing time across fusion approaches. Time values were measured on a standard laptop[5] with a precision down to the microsecond.

Figure 5.11, depicts the performance of each fusion variation type with respect to V_{max} values. Word wrapping W_x had no noticeable impact and smooth variation curves depicted similar behaviours to plain fusion. Both were therefore omitted for clarity purposes. Fragmentation speeds of 9.57 ms in average were measured for $V_{max} \approx 0.4$ which is 5ms faster than values measured by Kohlschtter and within the acceptable range. However, although rule based fusions depict relatively stable behaviour across threshold values, fragmentation time for plain fusion increases exponentially for $V_{max} > 0.6$. A further analysis of both algorithms revealed that, for rule based algorithms, most fusions occurred during initial block array iterations (section 3.4.2), while non-rule based fusions required an increasing amount of iterations to complete for large threshold values. Although $V_{max} \approx 0.4$ and $W_x \approx 80$ recommended by Kohlschtter do provide fragmentations at an acceptable speed, slicing systems in contrast require variable threshold values to be used based upon the sizing output of fragments selected (section 5.2.3).

Additionally, although the slicer implemented for the purpose of this research uses a-priori fragmentation with a fixed arbitrary fragmentation size (in order to also support third party collaboration requirements section 4.4), the fragmentation process could also occur on-demand if necessary. Since the desired granularity of slices is unknown prior to requests being submitted, the expected overall time performance of the algorithm (with a worse case scenario about 7 times the optimal one) in that case would also be difficult to determine in advance. Although DCF could certainly be easily parallelised, in the event clusters of machines are unavailable, a stable time performance across granularities would certainly be very desirable.

Conclusion: Although time performance measured achieves better results than expected, the increase in time necessary to process pages with respect to the V_{max} parameter, and hence sizing of fragments output, makes DCF an inadequate match

[5]CPU: 2.8Ghz Intel Core 2 Duo, Mem: 4GB 1067 MHz DDR3

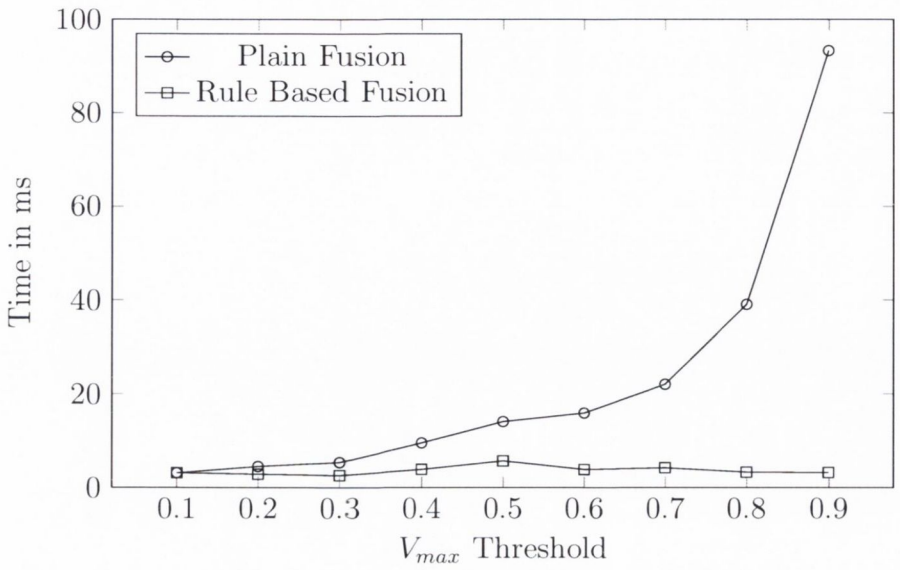


Figure 5.11: Plain and Rule Based Fusion Time Performance

for requirement TDR 3.a and hence fails to fully validate H1.d.

5.2.7 Greedy Fragmentation

As discovered in the previous section, non-rule based densitometric algorithms exhibit a sudden increase in fragmentation time per page for threshold values $V_{max} > 0.6$ which unfortunately makes the algorithm very unattractive within the context of slicers.

As explained and described in section 3.6, an alternative version of the original DCF algorithm, named Greedy densitometric content fragmentation, was designed as an attempt to address the time performance issues discovered in the previous section, as well as taking advantage of regional text density variation of pages to increase the accuracy of the algorithm. This algorithm was implemented and subsequently analysed as for previous algorithms.

As can be observed in figure 5.12, the greedy window-expanding algorithm reduces significantly the average time needed for block fusion per page, for high threshold values V_{max}. Although it is non-rule based, this algorithm depicts very similar time performance behaviour as its rule-based alternative with a performance increase

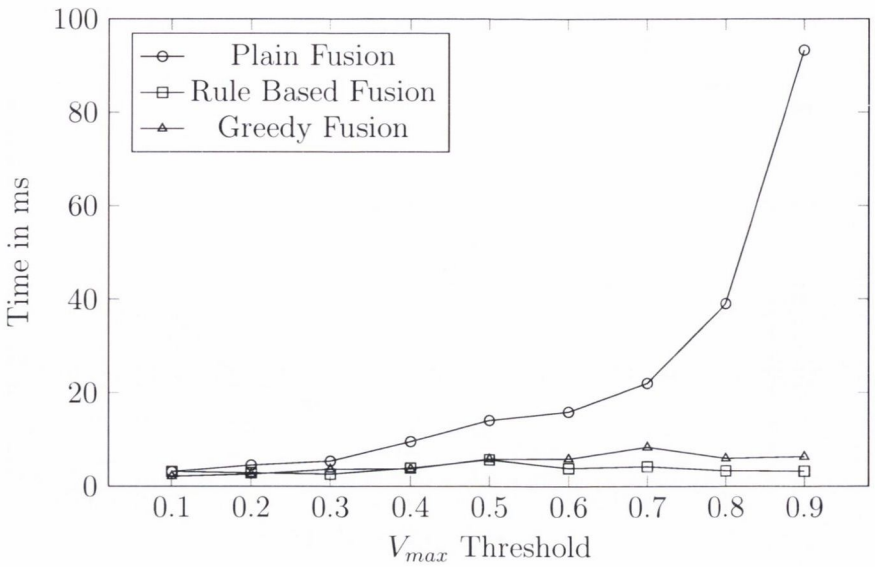


Figure 5.12: Greedy Fusion Time Performance

of 56% in average with respect to plain fusion and up to 89% improvement for threshold $V_{max} \approx 0.9$ (representing 10 million pages processed in 22.3 hours which is within acceptable boundaries established in section 3.3.3). Finally, slight accuracy improvements with both encyclopedia and news content types can also be observed in figure 5.13, (within the region $0.3 < V_{max} < 0.8$), which confirms the automated local threshold adjustment heuristic mentioned earlier.

Conclusion: Results obtained in section 5.2.6 for plain fusion DCF, depicted poor time performance for threshold values $V_{max} > 0.6$, which could be very detrimental within the context of a slicing pipeline if the sizing of slices were to be performed during the content fragmentation phase of the slicer pipeline. The new greedy DCF variation proposed in this section, on the other hand appears to improve both the time performance of DCF algorithms across fragment sizes (H1.d). Finally, results obtained across content types suggest the use of a variable threshold value can also provide minor improvements with respect to accuracy. Hence, although plain fusion DCF provides poor results with respect to time performance across various fragment output sizes, the use of a greedy DCF variation would appear to present a better alternative with respect to this requirements, if slice sizing is to be performed at the fragmentation phase of the slicer pipeline.

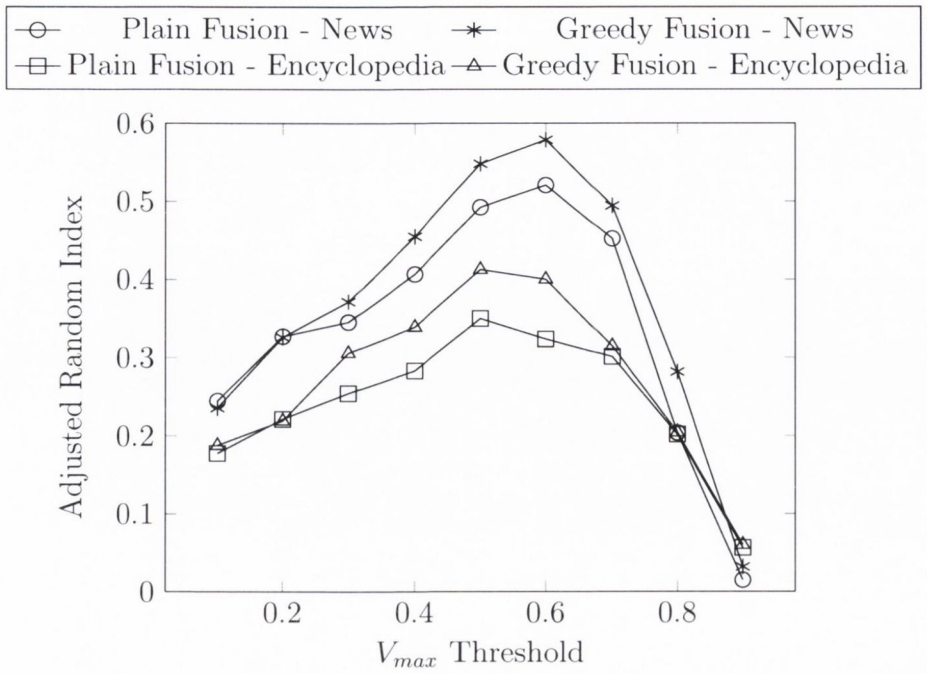


Figure 5.13: Greedy Accuracy

5.2.8 Parameterisation and Tuning of Greedy Algorithm

DCF algorithms can be used for many purposes using various parameter combinations. The parameterisation and tuning of the greedy fusion DCF algorithm offers the ability to maximise performance and behaviour depending upon different priorities (e.g. sizing predictability over multilingual correlation, or time performance over sizing predictability). As results obtained in previous sections, suggest fragmentation outputs can overall be predicted based upon different parameter settings (with varying levels of predictability for each fragmentation aspect considered), it should be possible to outline the expected fragment output obtained, based on combination of parameters provided and goals sought. This section hence consolidates and combines the overall results obtained in previous experiments. It attempts to provide guidance to the community with respect to the cross-cutting predictable effects, as well as trade-offs, of each parameter combination with respect to various fragmentation goal priorities (H1.e).

Sizing	Time Performance	Multilingual Correlation	Sizing Control	Recommended V_{max}
Small Sizing ($0 < V_{max} < 3$)	High			Towards Lover Values
Medium Sizing ($3 < V_{max} < 6$)	Medium		Low	Towards Lover Values
Large Sizing ($6 < V_{max} < 9$)	Low	High/Low - Very Sensitive	High	Towards Higler Values

Table 5.3: Parameter Table

In table 5.3, each performance combination (from low to high[6]) is highlighted with a colour scheme ranging from desirable (green) to undesirable (towards red values). As can be seen, low fragment sizing will produce the best results with high speed, high multilingual correlation and high predictable control over the sizing output of fragments. Medium sizing requirements however, will decrease the time performance and multilingual correlation between fragments, while also resulting in high standard deviation of fragment sizes. Although fragmentation speed will be decreased even further when choosing high sizing output, resulting fragments will possess similar sizes. In this scenario however, multilingual correlation between fragments will be highly sensitive (section 5.2.5), hence high fragment sizing parameters should be avoided if high multilingual correlation is required. For a given threshold, low wrapping values W_x will tend to provoke the measured effects upon time performance, multilingual correlation and sizing control sooner, and at the same time increase the volatility of the resulting performance, while high W_x values will have the opposite effect

5.2.9 Result Summary and Discussion

The analysis performed upon DCF algorithms confirms assumptions regarding the suitability of this algorithm with respect to specific slicing systems requirements such

[6]A high time performance indicates the algorithm will execute at high speed

as its ability to fragment without the need for interpreting the meaning (in the case of non-rule-based fusions) (TDR 3.e) or structure of tags (TDR 3.f) within pages.

It additionally provides the ability to fragment pages across languages (H1.c) and provides a predictable control over a wide range of fragment sizes (H1.a), cross-language accuracies and trade-offs based on parameters selected (H1.e).

However the densitometric approach to fragmentation was discovered to be highly content type dependent (H1.b), with higher accuracies achieved for news and encyclopaedia contents, suggesting an intuitive ranking of content type accuracy by degree of editorial control. A significant time performance decrease for large fragment sizes values was also discovered (H1.d) which makes this algorithm highly unattractive within the context of a slicing system (TDR 3.a).

Despite this drawback, a new greedy fusion algorithm, using an expanding window and automated local threshold value adjustments, provides a significant increase in time performances (up to 89%), which stabilises time performance across granularities, as well as providing higher accuracies. This new densitometric fragmentation technique improves the usage of this approach to fragmentation within the context of slicer systems. If this approach to fragmentation is to be selected for mainstream content slicing usage, further investigations should investigate and address any additional content type performance dependencies (section 6.3).

5.2.10 Consequential Design Requirements Refinement

The DCF analysis performed in this section validated the use of this approach to fragmentation with respect to most Technical Design Requirements (TDRs) presented in section 3.3.3. Nevertheless, three important limitations, with direct repercussions to this research were discovered.

Although results suggest this approach can indeed fragment open corpus resources independently of languages (TDR 3.b), in the event multilingual slicing is to be performed in parallel across identical documents in different languages, minor decreases in cross document accuracy will need to be taken into account. However, as the implementation of a multilingual slicer is considered out of scope for the

purpose of this research (section 3.3.3), the impact of this discovery on the prototype implemented is negligible.

The consequences related to time performance differences measured (TDR 3.a), with respect to different fragment size outputs, on the other hand are more significant. The Greedy DCF variation proposed by this research however did appear to stabilise the performance across fragment sizes, which reduces the impact of this discovery upon the implementation carried out for this research. Nevertheless, as is described in section 4.4, the requirement for the slicing process to enable the collaboration with third party institutions (DR 5) involved the decision to change the module responsible for the sizing of slices from the content fragment module to the slice generation unit of the slicer instead. Hence the impact related to the discovery of plain fusion time performance instability across fragment size is ultimately also negligible for this implementation.

The discovery of a DCF content type dependency however clearly limits the extent to which TDR 3.c can be fulfilled. Although, as described in section 2.4.3.5, DCF algorithms can process any resource available in XML format, this requirement alone will not determine the accuracy of the fragmentation. This content dependency must also be taken into account when selecting this fragmentation approach. Although several solutions tackling this limitation were proposed in section 5.2.4.2, an investigation of these is reserved for future work (section 6.3). It was therefore decided to instead temporarily narrow down the range of open-corpus resources, targeted for content/AHS agnostic slicing, to article based resources (news pages, encyclopaedia, tutorials etc.) in HTML format.

5.3 Design Requirement Implementation

5.3.1 Introduction

Chapter 3 presented a list of high level design requirements further extended and refined into technical requirements (section 3.3.2 and 3.3.3 respectively) for the implementation of a slicer prototype. This section therefore aims to provide the reader with a summary of how each of the implementation decisions taken in chapter 4 contribute to these design requirements. The following sections of this chapter will subsequently focus on the evaluation of the overall prototype implementation and its limitations.

Since many implementation decisions presented in this chapter affect several design requirements simultaneously, the decision was taken not to present these implementations decisions in the order each design requirement was presented to the reader in section 3.3.2, but instead to group these by themes. Figure 5.14 also provides a graph based depiction of these dependencies, summarising how each implementation decision contributed to each design requirement.

5.3.2 Low Cost Automated Slicing Service

DR 1 and DR 2 required the prototype, implemented for the purpose of this research, to provide a ***fully-automated*** open-corpus reuse solution. Apart from the initial provision, by independent AHS designers, of arbitrary repositories to be targeted by the slicer, the Slicepedia implementation presented in chapter 4, performs the reuse of open corpus resources, from the harvesting to the right-fitting and delivery of slices, without any human intervention. In particular, the use of i) a SaaS crawler (configurable programatically using an API), ii) fully automated NLP pipeline and content analysis algorithms as well as iii) content requirement requests and slice delivery interactions with AHSs performed programmatically via Restful interfaces, removes the need for any human intervention in the slicing process. The decision to use Restful interfaces in order to support the slicer/slice consumer interaction also enables Slicepedia to be ***accessed as a service*** by AHSs.

Finally, DR 7 also required the slicer implementation to be performed at ***low cost***. Following sub-requirement TDR 1, the vast majority of components used within Slicepedia were built using either open source tools or free SaaS services. Whenever this was not the case, free and open source alternatives were also provided to the reader.

5.3.3 On Demand Content/AHS Agnostic Slicing

Design requirement DR 2 required Slicepedia to be implemented using ***content/AHS agnostic*** techniques[7]. The content agnosticity requirement was supported through the selection of crawling and analysis algorithms which accomplish their task without the need for any a-priori knowledge of the content processed. More specifically, i) the 80Legs crawler for instance can harvest any document accessible through a simple URI[8], while ii) through to the support of the GATE platform, all NLP algorithms used within the analyser unit pipelines (with the exception of the DCF algorithm) perform their task independently of the source or selected format of content processed.

Additionally, iii) the DCF algorithm, selected to fragment native open corpus resources, was selected based upon its ability to process pages consisting of any xml type documents, theoretically regardless of domain, language or content type. However, as explained in section 5.2, a limitation related to this algorithm's performance with respect to various content types was discovered. Hence, for the purpose of this research, the range of resources, for which this slicer could be content/AHS agnostic, was narrowed down to any article type resource (News, Encyclopedia etc.). As mentioned in section 6.3, this limitation represents an important drawback. Hence if this approach to fragmentation is to be used for the purpose of content/AHS agnostic slicing, this restriction should be investigated further.

Furthermore, content agnosticity was further supported by TDR 2, which required the implementation choices made for the elaboration of this prototype to support

[7] narrowed down to any HTML content in English (section 3.3.2 and 3.3.3)

[8] http://wiki.80legs.com/w/page/1114616/FAQ

the ability for future slicer to perform multilingual slicing. For this reason, i) NLP algorithms in various languages, available for each task accomplished within GATE pipelines, were referred throughout chapter 4, while ii) the DCF algorithm was selected, as stated above, partly for its ability to fragment resources regardless of languages (TDR 3.b). The latter was verified in the experiment carried out in section 5.2. Furthermore, this requirement was also supported through iii) the decision to select the GATE NLP framework to implement the analyser unit of this slicer, due its capacity to provide conditional NLP pipelines (TDR 8) necessary for multilingual NLP analysis to occur.

The AHS agnosticity, as well as the ***on-demand*** characteristics of the slicer (DR 3), on the other hand were supported through the implementation of the slice generation unit. Although some a-priori analysis and processing of resources can be performed if necessary for optimisation purposes (section 4.2), both the provision of content requirements by AHSs and delivery of slices, is performed on demand. In other words, this unit only produces slices once it has received niche content requirements of slice consumers. The provision of content requirements by AHSs (expressed as a niche content requirements) only at run-time to the slicer service, means the slicer does not have any knowledge of either i) what AHS will be making such request, ii) according to what unique niche content requirement combination and iii) for what specific reuse purposes. As will be described in the following section, the AHS agnostic characteristic of a slicer however (including this prototype), will invariably always be limited to the set of broad content requirements supported by this slicer. Hence, while the content agnosticity of the slicer is supported through the selection of content analysis algorithms agnostic of the resources processed, the AHS agnosticity on the other hand, is supported through the slice generation unit which provides the on demand generation and receipt of niche content requirements.

Finally, the implementation of an on demand slice generation process through a three stage process (section 4.3.4.1), additionally contributes to supporting DR 6, which requires slice production process to be of type ***master/slave***. More specifically, the ability for a slice consumer to request slices via this three stage process, enables the

control over the slice generation process is to be handed over to slice consumers in priority. Each AHS can hence analyse the slicehits returned by the slicer and refine their request until the most adequate slicehits are returned by the slicer. Only then does the slice consumer request specific slices to be produced, according to selected slicehits.

5.3.4 Shareable Open Corpus Content Right-Fitting and Recomposition

DR 4 required Slicepedia to ***support the recomposition*** of slices produced, within independent AHSs through i) the correct selection and right-fitting of open corpus resources, as well as ii) their reuse via multiple search and delivery channels.

With respect to the right-fitting of resources, TDR 5 further refined that requirement by specifying the right-fitting should provide control over i) the granularity and ii) style of slices, as well as iii) providing adequate annotations supporting subsequent page adaptation by AHSs. As mentioned in section 4.6, Slicepedia provides a CAS enabling slice consumers to specify niche content requirements to which open corpus resources should be right fitted to. This CAS lets AHSs right fit resources with respect to the granularity of slices (sizing, focus), style (reading difficulty, format etc.) and annotations. As mentioned in the previous section with respect to AHS agnosticity, since the broad range of content requirements supported by a slicer, including annotation type required for AHS page adaptation, is closely linked to the set of possible AHSs, annotations provided by Slicepedia supported adaptation needs of the slice consumer used in the experiment carried out in section 5.4. These annotations were provided through the inclusion of a morphological analyser, verb group chunker and verb feature analyser within the semantic analysis pipeline section 4.3.3.5. Furthermore, the integration of slices produced by Slicepedia (which due to the open source of native resources they are produced from cover an open domain), with independent AHSs was supported through the use of linked data DBpedia domain anchors. This allows the slicer not to depend upon a predetermined AHS domain model and instead let each AHS connect its own model to a standard shared

open domain model.

Nevertheless, although the slicer prototype implemented for this research is indeed AHS agnostic, within the range of broad content requirement needs it supports, this limitation only lets it support a set of AHS slice consumers with predefined shared broad content requirement needs (as opposed to a unique predefined AHS with predefined reuse intentions and content requirements). In order to further extend this AHS agnosticity DR 5 hence required the generation of slices to be ***open and shareable*** between third party institutions. With respect to AHS agnosticity, this design requirement opens up the type and number of annotations (and therefore the set of broad content requirements which a slicer can support) available to a slicer, by reusing slicing data between independent institutions, and therefore truly serve an unrestricted set of AHS slice consumers. To achieve this objective, TDR 7 further required slicing data to be stored and accessible as RDF data. This was performed by the use of the Virtuoso triplestore mentioned in the previous section as well as with the implementation of a RDF converter module exposing both fragments of native resources processed as well as annotations produced as linked data to the community. The ability to share the process involved in the generation of slices between third party institutions was carried out by implementing the semantic analyser unit of the slicer as a regular slice consumer interacting with the slice generation unit. Both previous decisions enable any independent institution to query and analyse fragments produced by Slicepedia and publish their own annotations as linked data. The resulting set of cross-institutional fragment and annotations results in the form of a web of slices, aimed at increasing the volume of annotation and broad content requirements available to the AHS community. Web of slice data is thereafter accessible by any slice consumer through the use of contextual graphs specified in a SliceQuery object.

With respect to the second dimension of recomposition, the selection of slices was performed through the implementation of TDR 4 requiring Slicepedia to support keyword-based and conceptual selection of slices. These requirements were implemented by storing the slicing data within a Virtuoso triple store supporting

both SPARQL and full-text search. This enabled conceptual selection of slices, using SPARQL queries, based upon Dbpedia RDF based annotations assigned to fragments (TDR 7).

The search and delivery of slices was implemented based on TDR 6, refining design requirements to include a SPARQL, REST and cloud based interface to Slicepedia. While the SPARQL interface was available directly through the Virtuoso endpoint, the REST and cloud based interfaces were implemented using the Jersey API framework and an App Engine Pull Task queue client.

5.3.5 Flexible Architecture Integration

DR 8 required Slicepedia to be implemented using a flexible architecture enabling various components to be switched with each other. For this reason, Slicepedia was designed as a pipeline of successive components each processing resources forwarded to it independently of previous components (section 3.3.3.1). This overall pipeline architecture allows component implementations to be switched very easily depending on specific needs or evolution of individual state-of-the-art areas.

As depicted in table 5.4, the architecture was implemented with very straightforward integration interfaces. The first component interface involved with the external slicer prototype communication with slice consumers. This interface was provided through an abstraction layer using simple SliceQuery and SliceHit objects, representing slice requests and slices delivered respectively. This layer of abstraction enables any alternative storage mechanism to be chosen without having to change any implementation related to the slicer/slice consumer relationship and additionally reduces the complexity for slice consumers involved with SPARQL queries.

In order to increase flexibility, internal interfaces within the Slicepedia itself were also implemented. The integration between the native resource identifier module and the URI harvester module simply consist of lists of URIs, hence, if required, any additional crawler could easily be added to the pipeline with minimal implementation changes to the rest of the slicer.

Sinilarly, in relation to the resource harvester and analyser units, as mentioned in

Integration Interfaces		
Modules		**Interfaces**
Slice Consumer	Slicepedia	SliceQuery and SliceHits
Native Resource Identifier	URI Harvester Unit	URI List
Resource Harvester Unit	Analyser Unit	Local File System
Analyser and Slice Generation Unit	Triplestore	OpenRDF API

Table 5.4: Module Interfaces

section 4.2.2, resources harvested by the first component are stored within a *native resource repository*, which simply consists of a directory on Slicepedia's local file system. Hence, in a similar way as in the previous case, only a list of file paths pointing to each resource to be fragmented are forwarded between the two units, which minimises any integration effort necessary if internal components are modified.

Finally, SPARQL interaction between the open corpus resource analysers, slice generation units and the triplestore are all carried out using the OpenRDF API[9]. Since this API is free and open-source, this enabled Slicepedia to alternatively and seamlessly switch between various triplestore implementations. This flexibility in the architecture turned out to be useful when the necessity to switch between Sesame and Virtuoso triplestore implementations occurred (both supporting this API).

[9]http://www.openrdf.org/

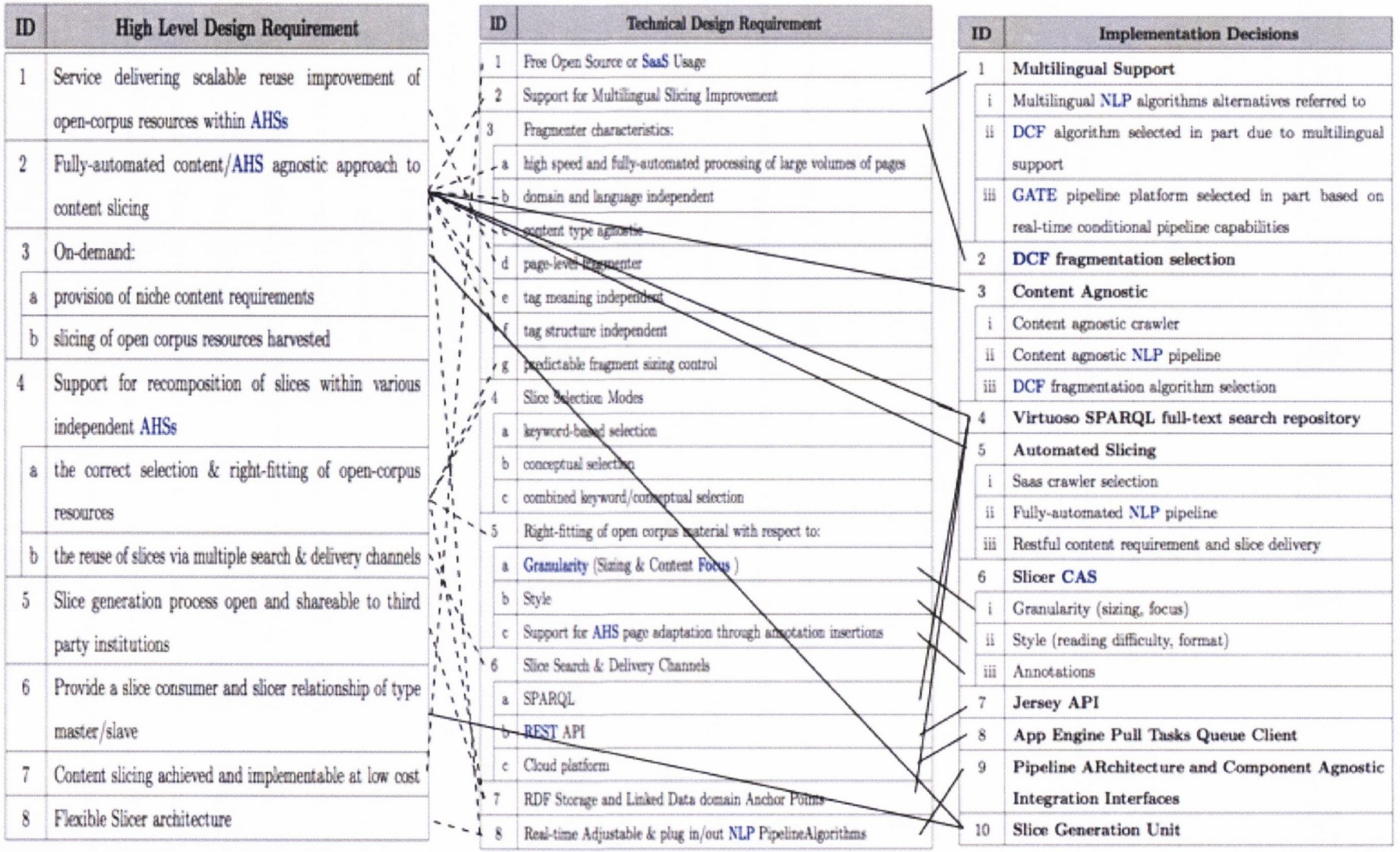


Figure 5.14: Implementation Decision Design Requirement Support

5.3.6 Conclusion

This section presented to the reader a concise summary of the implementation decisions involved in the development of the slicer prototype carried out for the purpose of this research, and their contribution to each design requirement. The following sections in this chapter will subsequently present trial-based evaluations carried out upon the overall slicer prototype with respect to aspects with as the reuse of open corpus resources and their recomposition within independent AHSs.

5.4 Content/AHS Agnostic Slicing Reuse Improvement Evaluation

5.4.1 Evaluation Objectives and Hypothesis

The previous two sections of this chapter focused upon evaluating individual components selected for the implementation of a slicer prototype (section 5.2), as well as describing how implementation decisions supported design requirements enunciated in chapter 3 (section 5.3). This section instead concerns itself with evaluating the primary objective of this research. In other words, the overall slicing approach to open corpus content reuse proposed as part of this research, with respect to the reuse improvement dimensions presented in section 1.2 is assessed. In order to perform this task, this evaluation compared four different approaches to open-corpus content reuse, namely i) direct reuse, ii) content specific reuse, iii) content/AHS agnostic reuse and iv) manual reuse.

As is presented in the following section, the overall procedure undertaken in this evaluation revolved around three major objectives:

i) The first consisted in evaluating whether an approach to open-corpus reuse using slicing techniques in ideal conditions (in other words content-specific slicing reuse with full prior knowledge of both the content targeted and reuse intentions) actually improves the reuse of these resources or not compared to direct or manual reuse. Any reuse degradation highlighted at this point, could only be due to the overall approach proposed by this research.

ii) The second objective of this evaluation, on the other hand, focused upon measuring the performance of a content/AHS agnostic slicing approach to open corpus reuse with respect to the other three alternatives mentioned above. In other words, assuming open-corpus reuse through slicing in ideal conditions does improve reuse, how does a content/AHS agnostic approach perform?

iii) Finally, content/AHS agnostic slicing was evaluated with respect to whether

such an approach could be selected as a replacement to the manual provision of content, produced specifically for a particular user task.

The objectives presented in the previous section are summarised here as three hypothesis.

Hypothesis:

H1) Content slicing, in ideal conditions, improves the reuse of open-corpus resources

H2) Slicing using a content/AHS agnostic technique does not present any major decrease in performance in comparison to content specific techniques

H3) Content/AHS agnostic slicing could replace the manual provision of content, produced specifically for a particular user task

5.4.2 Experimental Set Up and Methodology

5.4.2.1 Overall Methodology

Although the reuse of open-corpus material is ultimately aimed at recomposition within AHSs, this experiment focuses, as a first step, on evaluating the ***reuse of individual slices***. The assumption is that, in order for open-corpus content to be reused through recomposition with other slices, the quality of individual slices delivered must be guaranteed to AHS consumers. Any quality issues arising within individual slices would subsequently affect any recomposition produced that includes these objects.

Since content consumed by AHSs is ultimately presented to people, user experience should be considered a critical component of any content evaluation. The approach chosen for this evaluation was therefore to present a group of users with identical sets of open-corpus resources, reused using various techniques, within identical scenarios. This approach would provide direct comparison between reuse techniques and thus highlight any reuse differences perceived by users. Any differences between sets of resources could only be caused by reuse approaches evaluated.

Furthermore, the evaluation carried out as part of this experiment focuses upon measuring the ability of a slicer to ***improve the reuse*** of individual open-corpus resources, as opposed to its ability to identify relevant resources. Hence, and as will be explained in more details in section 5.4.2.5, the IR based open-corpus identification components of the slicers, which were evaluated were isolated for this experiment. This decision ultimately encapsulates the research question addressed by this experiment as follows: *Assuming a relevant open-corpus resource was initially identified, can the process of slicing open-corpus content improve its reuse?*

The reuse approaches presented in the previous section are described here in more detail.

Direct reuse of open-corpus content represents the equivalent of a traditional IR based open-corpus reuse strategy (section 2.3.4). Open-corpus resources in this context are reused and delivered unchanged, in their native form, as one-size-fits-all document objects.

Content specific reuse on the other hand, represents a semi-automated slicing approach to open-corpus content reuse. This approach represents a reuse scenario in which the designer of a slicer has full knowledge of the origin, structure and reuse purposes of open-corpus resources targeted for reuse. In these circumstances, the slicing process can therefore be designed in advance, in order to maximise the quality of slices produced, using algorithms specifically designed for selected repositories and reuse purposes. Slices produced by this approach, hence depict the ideal quality of content packages deliverable by a slicing approach.

Content agnostic reuse represents the slicing approach reusing open-corpus resources, fully-automatically, without any prior knowledge of either what repositories are targeted for reuse or for what specific AHS niche content requirements.

Manual reuse, finally was also considered within this evaluation for comparison purposes with previous reuse strategies presented earlier.

5.4.2.2 Experiment Metrics

As defined in section 1.2, throughout this research ***reuse improvement*** of open-corpus resources is defined with respect to the ability of an AHS to request such resources *i*) without any pre-determined conditions *ii*) as right-fitted content packages, *iii*) based upon a *variety* of possible AHS content requirement specifications. Although reuse improvement through *i*) and *iii*) are necessarily verified by the design and implementation of the prototype developed for this research, reuse improvements achieved through the right-fitting of open-corpus resources *ii*) still needs to be evaluated. The ability to specify what content to receive based upon a large variety of content requirement possibilities is pointless if the right-fitting of native resources isn't performed correctly. For this reason, reuse improvement measurements in this experiment focus specifically upon the effects produced by the right-fitting of native resources.

Measurements aimed at evaluating any reuse improvement of open-corpus resources, consist of two components namely, i) improvements due to *correct right-fitting* and ii) *quality decrease* of native resources as a result of right-fitting. Among other requirements, the ***right-fitting*** of open-corpus resources is required to be performed with respect to content granularity, including content focus, and to support AHS page adaptation through the insertion of annotations (section 3.3.3). The right-fitting should also produce SCI content (section 3.2). For these reasons, the right-fitting of content was evaluated based upon, its ability to correctly focus native resources, the quality of annotations inserted (using standard precision and recall metrics), as well as the impact upon the reuse caused by various slice SCI accuracies.

Quality decrease, on the other hand, more specifically refers to the readability and visual structural coherency of content (i.e parts of original page not merged erroneously with menus for example). This research defines readability as being composed of two sub-components, namely i) the ability of content to be easily understood, and ii) the extent to which the flow of reading is broken. The second component refers directly to the notion described by [Ganguly2011], in other words, the extent to which pronouns, for example, referring to earlier subjects missing from

a fragment, affect the ease of reading and comprehension of a text.

Since ***production cost*** is dependent upon either local prices of manual labor, or individual server specifications, an estimation of production time was considered instead a better proxy for production cost measurements. Moreover, time measurements would additionally enable future performance comparisons. For this reason, the time required to produce content batches was measured as opposed to direct costs.

Finally, this experiment additionally measured the content produced, using various approaches, based upon its ***suitability*** with respect to a determined task performed by a group of users. As will be described in section 5.4.2.4, within the context of this evaluation, suitability performance refers to the ability of content produced to support a task performed by individuals. As described in the following section, the use-case of student language assessments was used for this experiment, hence for this reason, the number of mistakes and time taken by users to perform assessments were used for this measurement.

Finally, qualitative measurements for each measurement dimension presented earlier, using questionnaires, as well as overall reuse appreciation of slices by users was also used as data supplementing quantitive results.

5.4.2.3 Content Specific Slicer Implementation

The slicer implemented for the purpose of this research was required to use a flexible architecture (DR 8). Section 3.3.3.1 described how Slicepedia processes pages using a pipeline architecture, allowing components to be replaced and switched with improved versions, as the state-of-the-art in each field progresses. For the purpose of this experiment, this requirement enabled components of the content/AHS agnostic slicer to be replaced by others in order to produce a content specific slicer.

Both implementations share common components but differ particularly with respect to the fragmentation algorithm used (see section 3.3.2). For the purpose of this experiment, the English Wikipedia encyclopaedia[10] was selected as the repository

[10]http://en.wikipedia.org/wiki/Main_Page

target for the content specific slicer. As mentioned in section 2.3.4, a content specific slicer by definition is built specifically for an arbitrarily chosen resource repository, using hand-crafted rule-based algorithms. The fragmentation process of Wikipedia pages hence should be based upon a set of pre-defined rules. Fortunately, the Java Wikipedia Library (JWPL) library[11], developed by Darmstadt university, was built specifically to achieve this task. Hence the densitometric fragmentation algorithm used within Slicepedia was switched with this approach. In order to produce fragments, JWPL requires Wikipedia pages to be initially harvested in XML format, converted and stored in a mySQL database[12]. The Java library can subsequently be used to extract specific sections of selected pages, with perfect accuracy and removed from any boilerplate. Because, any boilerplate content present in Wikipedia pages is automatically discarded by JWPL rules, the boilerplate removal algorithm used within Slicepedia was removed. It is worth clarifying at this stage that, while the two slicers used within this evaluation differ with respect to their content specificity/agnosticity characteristics, both on the other hand, share common characteristics with respect to the AHS agnosticity, since they both use the slice generation unit module (section 4.3.4.1) to receive unknown niche content requirements of AHSs. Hence, in this experiment, both slicers partially fulfil the AHS agnosticity criteria (section 3.3.2) in the sense that the specific AHS niche content requirements, which they must serve is unknown at design time. However, as will be pointed out in the next evaluation (section 5.5), in both cases, the reuse purpose scenario of slices produced is known in advance.

As specified in section 4.3.3.1, at the time this experiment was carried out, the Sesame triplestore had been selected for the Slicepedia implementation. Hence both slicers used a Sesame, as opposed to the Virtuoso triplestore presented in section 4.3.3.1. As pointed out in section 3.3.2, when targeting pre-defined open-corpus repositories, pages targeted can possess valuable indications related to the domain or meaning of content held within each page. Wikipedia pages are each classified within overall subject categories. This information was hence also retrieved when extracting

[11] http://code.google.com/p/jwpl/

[12] http://www.mysql.com/

fragments and stored within the triplestore. With the exception of Wikipedia categories, both slicers offer the same CAS to slice consumers.

5.4.2.4 Reuse Vehicle Selection

As explained within the section 5.4.2.1, this experiment aimed at evaluating various content reuse approaches, by presenting users with slices in identical circumstances, The scenario selected for this experiment should hence satisfy the following ***reuse vehicle requirements***: i) It should present users with a set of similar repeatable tasks, ii) based upon real-life needs of users. In order to avoid any possible interference with the content being evaluated, the reuse vehicle should iii) keep interface complexity to a minimum and iv) involve no recomposition of slices. Moreover, fully automated slicing of open-corpus resources might affect v) the understandability, reading flow and vi) annotation quality of slices produced (section 3.3.3). Hence, any content evaluation task should additionally be selected based upon its dependency with respect to these two properties. Finally, the selected evaluation scenario should also vii) provide the ability to measure easily the suitability of content provided to users based upon the task performed (section 5.4.2.2).

For all these reasons, a language e-assessment use-case application, of type and inspired from the use-case scenario presented in section 3.3.2.1, was selected for this experiment.

Language e-assessment scenario

> Suppose Alice wishes to improve her grammar skills in Portuguese and decides to use a portal specifically built for this purpose. The system provides e-assessments consisting of traditional gap filling exercises as presented in figure 5.15. It provides Alice with a list of various languages λ, grammar skills ξ and reading level difficulty ρ to choose from which she selects accordingly to her training needs. So as to sustain learner motivation, the portal additionally provides the ability to select topics of interest, among a large list θ, which training exercises should also cover. Whenever Alice starts her training, the system searches for resources on the web fulfilling the combined requirements

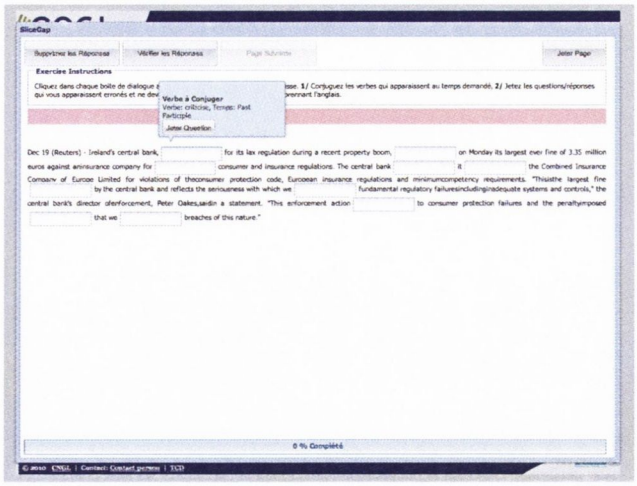


Figure 5.15: SliceGap Interface for French Native Speakers

$\Sigma(\lambda, \xi, \rho, \theta, \mu)$ and converts these into grammar e-assessments. The system continuously records the sets of mistakes μ performed by Alice and includes this additional variable to its content requirement combination Σ in order to address the required subset of grammar points of immediate importance. As Alice progresses, the set of requirements Σ evolves and so does the content supplied. The portal can supply Alice with as much content as needed for her training. This use-case supporting a wide range of diverse user requirements, impossible to predict in advance, represents a very good example of the type scenario, which a content/AHS agnostic slicer could theoretically support.

A reuse vehicle, called Slicegap, supporting this use-case was hence implemented as follows. Slicegap presented users with series of traditional gap filler exercises consisting of open-corpus resources reused using various alternative approaches (section 5.4.1). Slices supplied to Slicegap are provided with annotations identifying each verb chunk within the content. Annotations are assigned attributes consisting of the infinitive and tense of each verb chunk. Verb chunks annotated within the slices supplied are then removed by Slicegap and replaced by gaps consisting in HTML input boxes. For each gap, users are then presented with tooltips containing pairs of infinitives and tenses corresponding to each verb missing, which they are required to conjugate accordingly. Answers provided are then compared to the

original verb chunks supplied within each slice and users are assigned a score for each specific grammar e-assessment. Since, the overall aim of this experiment consisted in presenting a group of users with the exact same set of resources (section 5.4.2.1), personalisation characteristics of the use-case scenario presented above were ignored.

Although not perfect, this reuse vehicle provides an adequate trade-off between all reuse vehicle requirements presented above in this section. Slicegap e-assessments provided the ability to present users with various slices through a set of identical repeated tasks (requirement i), based upon real-life linguistic assessment needs of individuals (requirement ii). Because the selected use-case scenario is performed within the context of language learning, this offered the opportunity to invite users with varying levels of competency in the English language to perform the experiment. Hence, understandability of content presented to users, as well as reading flow could be measured depending upon the level of proficiency of each user (requirement v). Additionally, since the task performed by users is directly dependent upon the annotations produced (requirement vi), any quality decrease of annotations would provoke important assessment errors resulting in such e-assessments being pointless and time consuming to users. This use-case scenario also offered a straight forward way to measure suitability of content provided for such a task (requirement vii) by measuring time and error-rate differences performed by users between each content batch. Hence, within the context of this use-case scenario, suitability performance refers to the ability to correctly assess an individual. Finally, since e-assessments can be performed using individual slices, without the need for any recomposition, this additionally offered the ability to evaluate each piece of content presented to users individually (requirement iv), as part of a very simple interface (requirement iii).

5.4.2.5 Content Batch Creation

This section describes the process involved in creating the eight different content batches subsequently used throughout this experiment as described in figure 5.16. Section 5.4.2.2 specified that this experiment measures reuse improvement of open-corpus material with respect to the right-fitting process performed by a slicer. However, reuse improvement measurements could be affected by the relevancy of pages identified for reuse. As mentioned in section 1.1, the identification of *relevant* web resources by a slicer is considered out-of-scope of this research and instead assumes the resources identified by the chosen IR component are relevant to the needs of AHS slice consumers. For these reasons, it was necessary to isolate this component of the slicer pipelines used for the purpose of this experiment. Moreover, since this experiment aims at comparing four different open-corpus reuse approaches (including a manual approach) using an identical and truly random set of native resources, requesting independent users to arbitrary select relevant web resources of their choice, appeared to be the best option available. This strategy ensured the designers of this experiment had no control over the set of pages targeted for reuse, but nevertheless guaranteed a set of relevant pages could be used across the four reuse strategies described in section 5.4. Hence, prior to conducting the experiment, a group of five independent English teachers, from two different language learning schools[13], were asked to each arbitrarily select nine pages of their choice from the web (combined total of 45 pages) and to prepare parts of these resources for inclusion within the Slicegap reuse vehicle. This manual content production task deliberately involved only the repurposing of existing content (in the spirit of educator repurposing activities described in section 2.4.2). This decision aspired to replicate a manual content production scenario (used as a baseline) with minimal production time requirements. The assumption was that any content authoring activities would always require more time to produce.

As mentioned in section 4.3.3.4, due to a content type dependency of DCF algorithms,

[13]English Language Tutorial Academy (E.L.T.A.), and the House of Education in Language and Linguistic Observatory (H.E.L.L.O.)

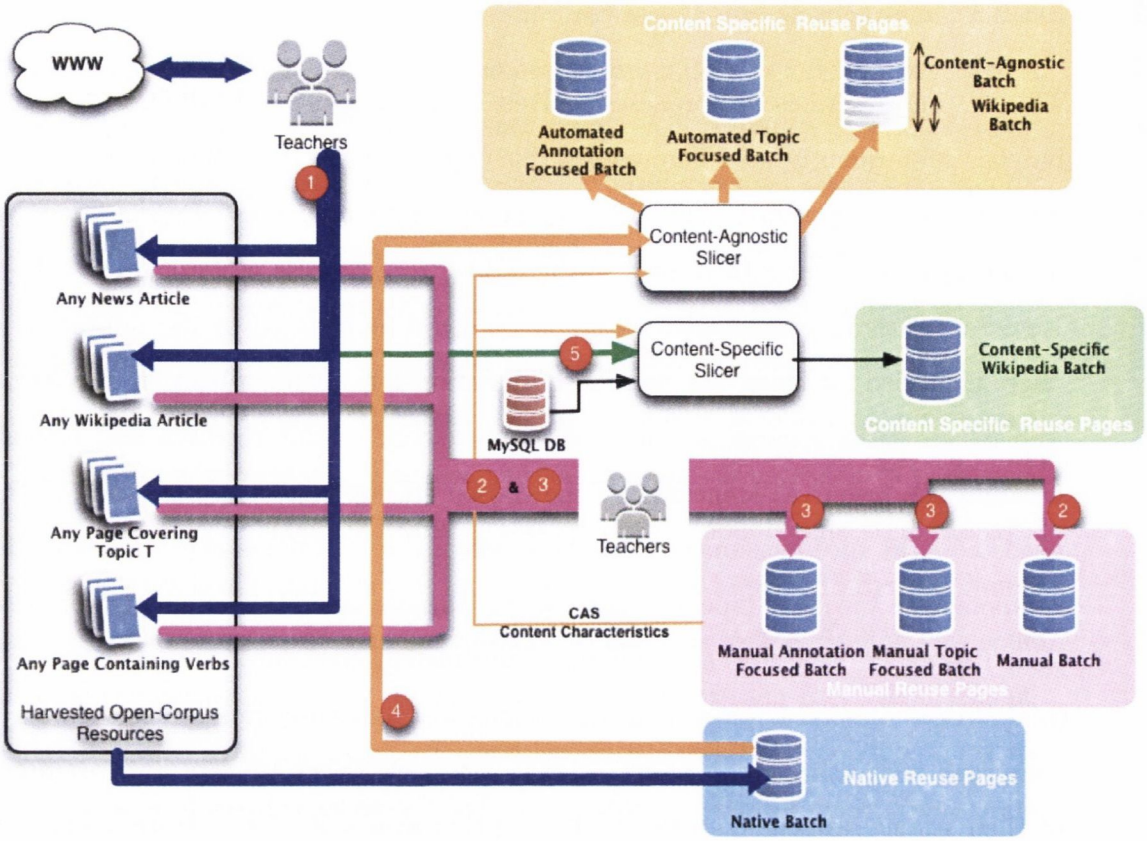


Figure 5.16: Content Batches Production

discovered in the previous experiment, it was decided to limit the range of open-corpus resources, targeted for the purpose of this research, to any HTML resource originating from either news or encyclopaedia websites. For this reason, the set of pages considered for this experiment were purposely restricted to this subset of the WWW.

Step 1: Page Identification

English teachers were first asked to arbitrarily select four sets of pages. Any page, constituting the first two sets, could be selected as long as they consisted of Wikipedia and news articles respectively. The third and fourth set of pages could be of any source as long as it covered an arbitrary concept γ (third set), or contained a set of ten verbs conjugated at an arbitrary tense τ (fourth set). The concept *human rights* was chosen for this tasks, in order to be accessible

to as wide an audience as possible. This enabled users to accurately judge the scope of the content presented. The entire set of pages identified were subsequently used as ***Native Content Batch (NCB)***.

Step 2: Content Extraction and Annotation

Teachers were then asked to select fragments, of any size, from the first two sets of pages harvested, which they felt could be adequate for English grammar exercises, and annotate each verb present within these extracts with the relevant tense and infinitive to produce ***Manual Content Batch (MCB)***.

Step 3: Content Focus

Finally, they were asked to extract fragments from the third and fourth sets of pages harvested, by ensuring these fragments were correctly focused respectively upon verbs annotated with tense τ and content covering individual concept γ (content not about this concept was discarded). These tasks resulted in the creation of ***Manual Topic Focused Content Batch (MTF)*** and ***Manual Annotation Focused Content Batch (MAF)***.

Step 4: Content Agnostic Slicing

The first two sets of open-corpus resources identified by teachers (both part of MCB), were harvested in their native form and then sliced by Slicepedia (using the content/AHS agnostic implementation described in detail in chapter 4) to produce content batch ***Content-Agnostic Batch (CAB)***. This content batch represents a true set of open corpus resources since the origin and structure of pages selected were unknown in advance of slicing (with the exception of Wikipedia pages). CAS characteristics (sizing, style, topics etc.) of fragments manually produced and arbitrarily chosen by teachers, were used as arbitrary niche content requirement parameters for any automated slicing that occurred for the purpose of this experiment. Native pages used to create content batches MAF and MTF were also sliced by the content/AHS agnostic slicer, using the same concept and annotation focus content requirement parameters to produce ***Automated Annotation Focused Content Batch (AAF)*** and ***Automated Topic Focused Content Batch (ATF)*** respectively.

Step 5: Content Specific Slicing:

The set of Wikipedia resources, arbitrarily identified by teachers, was then matched to those contained in the mysql database (see section 5.4.2.3) and sliced using the content-specific slicer to produce batch ***Content-Specific Batch (CSB)***. As mentioned previously, measurement differences across reuse approaches should be performed over the same exact set of native resources sliced. For this reason, the subset of fragments produced from Wikipedia pages within batch CAB is denoted as content batch ***Wikipedia Content Batch (WCB)*** and used when making direct comparisons between content-specific and content agnostic slicing.

Step 6: Slice Conversion:

Slices present in all batches were subsequently converted into grammar e-assessment pages.

With respect to the reuse approaches outlined in section 5.4.1, content batches NCB and CSB respectively represent direct and content-specific reuse of open-corpus resources, while batch CAB, AAF, ATF and WCB represent content/AHS agnostic reuse. Finally, as content batches MCB, MAF and MTF were produced manually by independent users, these batches represent the manual reuse approach.

5.4.2.6 User-Trial Scenario

The entire user-trial experiment was available online to the public with the interface available in English, Spanish and French. Native and non-native speakers were invited to select their interface language of choice and specify their level of ability in the English language. A total of 41 users across 7 different countries performed the experiment in its entirety. Most of these users performed the experiment using the English interface (en=66%, non-en=34%) and rated themselves as native or advanced English speakers. Based on the ratings, the users were divided in two categories consisting of Experts (E) (Native Level=46%, Advance Level=17%) and Trainees (T) (Intermediate=29%, Beginner=7%) both respectively representing a total of 63% and 37% of users.

The entire set of tasks was required to be performed within a single session without any interruption. Task completion time in addition to user interactions were tracked throughout each activity. Users were asked to perform successively the same activity using resources randomly selected from each content batch (section 5.4.2.5). In order to balance any effect of order bias, content batches were presented to users according to a Latin square design distribution and each user was presented with at least one resource from each batch. A unique colour was assigned to each content batch and users were unaware which was being presented to them. Each original page (regardless of what content batch it belonged to) was only shown once to each user. Users were asked to complete the e-assessment presented and subsequently answer a questionnaire. Each question was answered using a Likert scale, ranging from zero (strongly disagree) to ten (strongly agree). A scale with no mid-point was deliberately used to enforce user preference however slight. The order of sentiment in the scales was also randomised between questions in an attempt to ensure that users were genuinely attentive when answering each question. Finally, they were asked to order a set of colours, corresponding to each content batch presented, based on their perceived quality.

The tasks repeatedly performed upon each content consisted of a set of traditional language learning activities, encountered in most language learning textbooks. The

activity was divided in three individual tasks:

User Task 1:

The first task required users to read the text presented to them in Slicegap and fill in any blanks encountered (there were 10 gaps on average per page) with the appropriate verb and tense specified at each occurrence (see section 5.4.2.4). If users felt slices were unreadable or inappropriate for this exercise, a *Trash Slice* button could be pressed to select another page.

User Task 2:

The initial task did not necessarily require users to read the entire content presented to them (only the sentences containing gaps). For this reason, users were then asked to summarise what they felt the entire slice presented to them was about (as in traditional textbooks) in their own native language.

User Task 3:

Users were finally asked an additional set of questions, which assessed the focus appropriateness of each slice.

5.4.3 Results

The following sections present a summary of the findings observed throughout this experiment in relation to each hypothesis stated in section 5.4.1. The number of times the Trash Slice button (section 5.4.2.6) was clicked throughout the experiment was statistically insignificant. For this reason, it is not further discussed within these results.

5.4.3.1 H1: Slicing Reuse Improvement

As pointed out in section 3.2, maximising the reuse potential of a previously published resource requires, among other things, the ability to **right-fit** such a resource and convert it into ***Self-Contained and Independent (SCI)*** content. As explained in the glossary, SCI content is concise in the sense that it does not contain any superfluous content, unnecessary or detrimental for the proper reuse of such resources. Hence, users were asked directly, for each resource presented, whether *in addition to the main content, a lot of material displayed on the page was irrelevant to the task (such as advertisement, menu bar, user comments..)*. Figure 5.17 presents the results obtained over a Likert scale ranging from one to ten (section 5.4.2.6). Content batch NCB, containing pages in their native form, achieved the worse score with a mean equal to 5.67 while the content-specific (CSB) and content/AHS agnostic (CAB) content batches achieved much better scores ($CSB = 1.76$, $MCB = 2.13$, $CAB = 2.53$) with paired t-tests confirming results are indeed significant ($p = 0.001$). When comparing content-specific (CSB) and content/AHS agnostic (WCB) approaches applied upon an identical set of pages (the Wikipedia subset see section 5.4.2.5), a mean difference of 0.879 was measured ($p = 0.015$). This result suggests that, although the content/AHS agnostic approach to slicing did achieve a very good performance with respect to its content-specific equivalent (91% similarity), users did notice a minor performance decrease in its ability to remove superfluous content within native resources. Differences measured between MCB and CAB batches were insignificant ($p = 0.423$) which would indicate that content automatically produced achieved similar performance to content manually produced.

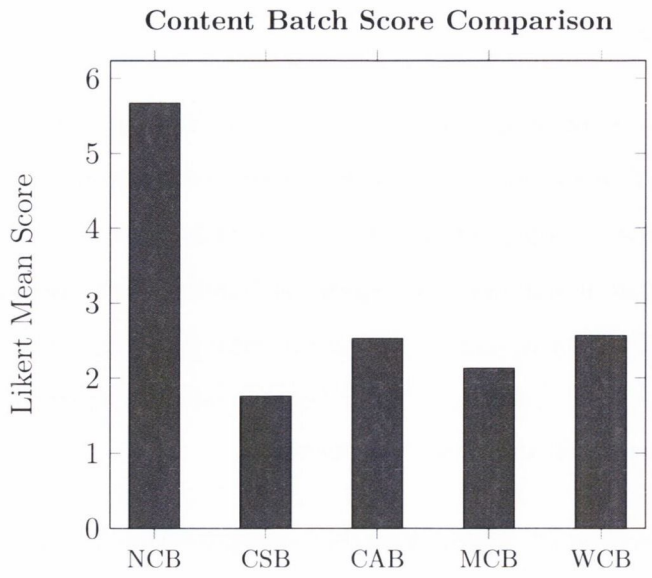


Figure 5.17: Superfluous Content

In order to determine the ***impact*** such a difference had upon the task accomplished by users, contestants were asked whether *the level of irrelevant material (e.g: menus, advertisement, user comments etc.) present in the content prevented me from pursuing this task effectively.* NCB resources again achieved the worse score (3.43) versus content-specific ($CSB = 1.48$), manual content ($MCB = 1.88$) and content/AHS agnostic ($CAB = 1.64$) batches (with paired t-tests confirming these results are statistically significant ($p <= 0.01$)). These results, indicate poor right-fitting of original resources impacts the reuse of such content. This observation was confirmed when measuring the average time, per answer provided by users, necessary to complete the exercises. As can be seen in figure 5.18, users in average required 88% more time to complete the exercises using resources in their native form (NCB) than they did for right-fitted resources with no superfluous content(CSB).This observation appears to confirm the assumption stated by [Lawless2009] (see section 1.1), whereby *there is an inverse relationship between the potential reusability of [...] content and its granularity.* The reuse of one-size-fits-all open resources is improved when removed of any superfluous content such as navigation bars, advertisement etc. Finally, differences in results measured between CSB and WCB for both the question and average time measured were insignificant ($p = 0.77$ & $p = 0.58$). This result indicates

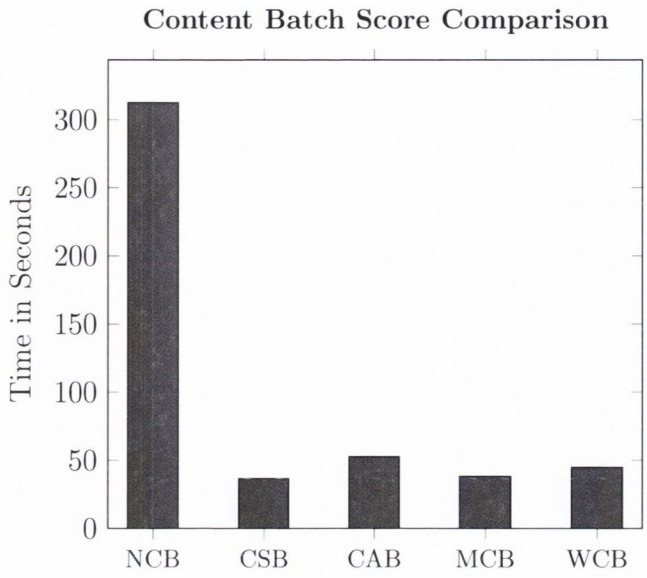


Figure 5.18: Superfluous Content Impact

that although a decrease in SCI content conversion performance of original resources for content/AHS agnostic slicing was noticed by users, this minor performance decrease didn't appear to have any impact upon the reuse of the content produced. Differences between MCB and CAB batches, were also statistically insignificant ($p >= 0.278$). Nevertheless, when plotting together both the superfluous removal performance (figure 5.17) and time taken to perform each exercise (figure 5.18), results obtained per batch appear to correlate with each other across the two measurements, which reinforces the indication that the correct removal of superfluous content does have an impact upon its reuse.

As mentioned in section 3.2, the ability to ***focus resources*** in their native form into slices containing only parts of original documents about a specific topic or set of requested annotations, is important with respect to the capacity of a slicer to deliver information objects to various AHSs, matching specific content requirements and use cases. Content batches MTF and ATF represent such a case of original resources being topic focused. When users were asked whether they *felt parts of the content presented were about the topic 'human rights'* , the manually focused content batch (MTF) obtained the best score with a mean equal to 9.09 versus the content/AHS agnostic automated equivalent (ATF) which obtained 8.55. A p value

of 0.82 suggests this mean difference between manually and automatically generated contents is insignificant. This result appears to suggest the content/AHS agnostic slicer was capable of automatically identifying which parts of the documents did refer to specific topics. Additionally, when asked whether *a large amount of content delivered was not related to [this] topic*, the manually focused content again achieved the best score in comparison to the content/AHS agnostic approach with a mean equal to 3.07 and 3.86 respectively for MTF and ATF. The paired t-test($p = 0.159$) suggests once more that the difference measured between the manual and automated approach is insignificant. As a sanity check, the answers for the same question asked for native content were also measured. Since pages within content batch NCB are presented as entire documents, they are necessarily unfocused; hence a very poor focused performance should be measured. As expected, a mean value equal to 6.43 for NCB suggests users felt this native content was not correctly focused. When asked the reverse (ie:*all the content presented was related to [this] topic*), the results lead to the same conclusion with ATF and MTF obtaining respectively 7.36 and 8 ($p = 0.411$), while NCB received a score of only 1.23. These results appear to indicate that not only could the content/AHS agnostic slicer correctly identify parts of the document on a particular topic, it could also minimise the amount of unrelated content delivered, and offer slices with boundaries tightly confined to that specific topic. Moreover, paired t-tests ($p = 0.159$ & $p = 0.411$) do suggest content focused automatically using the content/AHS agnostic slicer achieved similar scores to its manual equivalent.

Finally with respect to content/AHS agnostic content being correctly focused upon annotations requested, a similar pattern can be observed. When asked whether *apart from any clutter information (advertisements, menus etc.) displayed within the page, I felt a large amount of content delivered to me was un-necessary for the grammar task being performed*, content manually focused upon annotations (MAF) achieved the best score (3.53) with the automated approach presenting a minor difference (3.73) considered insignificant ($p = 0.783$). A sanity check with NCB again obtained the worse score of 5.87. These results indicate the automated content/AHS agnostic slicer could correctly focus native resources upon annotations requested

with a performance similar to its manual equivalent.

In order to measure the ***accuracy of annotations*** delivered to slice consumers, a sample set of identical pages annotated by teachers (section 5.4.2.5) was compared to those produced automatically by the content/AHS agnostic slicer. Individual manual annotations, obtaining the largest agreement among teachers, were used as a golden standard. Precision and recall was calculated for the annotations produced by the content/AHS agnostic slicer as well as for each individual manual annotator. An F score equal to 0.88 (precision=0.86, recall =0.92) was measured for the content/AHS agnostic slicer. This result indicates the annotator identified most items to be annotated, while producing a minority of errors during the process in comparison to human annotators. This high correlation between automated and manual annotations appears to confirm the quality of annotations produced by the automated content/AHS agnostic approach to slicing. Additionally, when performing the same measurement for each human annotator individually, with respect to the golden standard, an average F score of 0.77 (Precision=0.89, Recall=0.68) was obtained. This result suggests that although the quality of annotations produced by humans was higher than the slicer, some disagreements between teachers did occur; suggesting mistakes produced by the automated approach could be subject to interpretations as well. Additionally, the recall score indicates that many human annotators missed several annotations in comparison to the automated approach.

As described in section 3.3.3, the process of structurally fragmenting original resources into individual pieces and subsequently merging selected fragments together to form slices, can lead to ***structural incoherencies*** within the final content delivered, ultimately resulting in an overall **decrease in quality**. For this reason, users were asked directly if *parts of the original web page presented were merged inappropriately together (eg: end of menu merged with start of a paragraph)* (Q1). Table 5.5 depicts the results obtained per content batch. As can be observed, very good performances were measured across all content batches, with t-tests estimating any differences encountered as insignificant. These results suggest that the process of slicing resources didnt produce any significant increase in structural incoherencies in comparison to

		Mean	Comparison	Mean Dif	p
Q1	NCB	3.38	NCB vs CSB	0.25	0.734
	CSB	3.13	NCB vs CAB	0.28	0.662
	CAB	3.65	CSB vs WCB	0.688	0.357
	MCB	3	MCB vs CAB	0.65	0.463
Q2	NCB	8.19	NCB vs CSB	0.313	0.608
	CSB	8.5	NCB vs CAB	0.531	0.479
	CAB	7.65	CSB vs WCB	0.75	0.118
	MCB	7.78	MCB vs CAB	0.13	0.868

Table 5.5: Structural Coherence

the original content in its native form. Moreover, a mean difference of 0.688 observed between CSB and WCB was considered insignificant. This would indicate that, although measurements for the content-specific slicing approach were higher than for content/AHS agnostic slicing, these results do not provide enough evidence to suggest content/AHS agnostic slicing leads to any significant decrease in structural coherency of slices produced. When asked the reverse (Q2), opposite values were also measured, with similar t-tests, which confirms these observations.

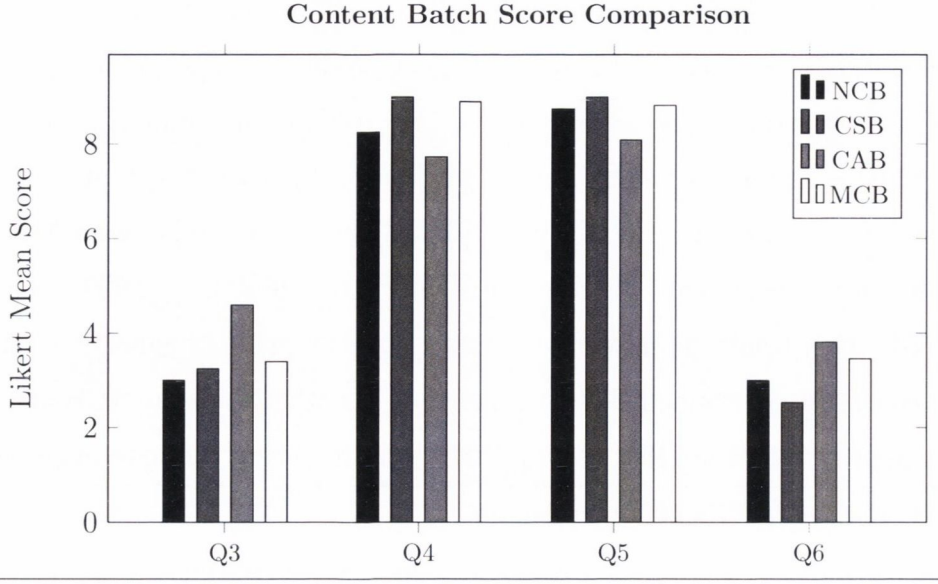
While previous measurements aimed at estimating any decrease in quality of the content delivered from the point of view of visual layout, the ***readability of content*** produced for reuse is of course critical. As mentioned in section 5.4.2.2, readability is decomposed into i) the ability of content to be easily understood, and ii) the extent to which the flow of reading is broken. Figure 5.19, depicts the results obtained for the first component measured. The first pair of questions aimed at asking directly to users their opinion with respect to how easily content presented to them could be understood. The second pair relates to measuring any impact upon the difficulty in summarising content. As can be seen, results obtained across content batches are very similar. While native (NCB) and content-specific slicing (CSB) approaches interchangeably achieved better results across questions, content sliced using the content/AHS agnostic approach (CAB) constantly performed lower

than the latter two. Paired t-test between CSB and WCB content however estimated any mean differences measured between content produced using content-specific or content/AHS agnostic slicers as insignificant ($p > 0.06$ for all questions). Differences measured between content batch CAB and MCB were also insignificant. An exception for question Q3 however occurred, in which a difference of 1.6 between NCB and CAB, which was significant for confidence intervals of 95% ($p = 0.028$), could be measured. This result, along with the fact that content/AHS agnostic approach (CAB) constantly performed slightly lower that the other than the NCB and CSB batches appear to indicate that content/AHS agnostic slicing can have minor effects upon the reading quality of content sliced.

As a sanity check, and in order to make sure all sentences within content presented were read (regardless of whether they contained gaps to be filled), users were asked to summarise the meaning of the content presented to them. Among all the summaries provided by users across content batches, only less than 5% were rated as weak. Despite being on topic, these answers were too short (less than 5 words) to determine whether the overall meaning of the content was correctly understood or not. Nevertheless, the overall quality of summaries submitted clearly supports previous results obtained.

Since close to a third of users, who participated in the trial where non-native speakers, a Pearson correlation analysis between answers submitted to each question and contestant category (section 5.4.3), was also performed. Correlations measured for each variable combination ranged between -0.2 and 0.2, which indicates no direct correlation between answers provided and the level of users. Additionally, when users were asked whether they felt *their language skills had any negative impact upon [their] ability to correctly summarise content*, means measured across content batches were all inferior to 2.30.

With respect to the second component of readability (reading flow), there exists, to the best of the authors knowledge, no automated mechanism to reliably estimate whether pronouns or topics mentioned within a fragment referred to earlier subjects mentioned in omitted parts of the original content. Hence users were asked directly (Q7) whether



Q3 I felt this content was difficult to read

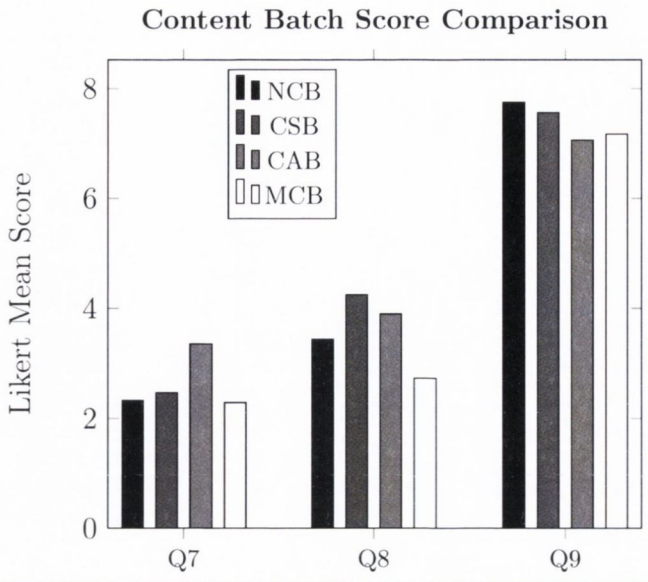
Q4 I felt I could understand easily what this content was about

Q5 I felt it was easy to provide summaries of the content

Q6 I felt the understandability of the content (as opposed to my language skills) had a negative impact upon my ability to correctly summarize the content

Figure 5.19: Understandability

this situation occurred or not for each content presented to them. Results depicted in figure 5.20, indicate NCB obtained the best results (for Q7), which is expected since the entire original resource was presented to users. The content-specific (CSB) or manual batches (MCB) presented no statistically significant differences with its native alternative (NCB) ($p = 0.685$), while results for the content/AHS agnostic approach (CAB) on the other hand, presented a small statistically significant decrease with respect to NCB (mean dif =1.03, $p = 0.037$) as well as MCB (mean dif =1.07, $p = 0.033$) for confidence intervals of 95%. This result suggests content/AHS agnostic slicing can indeed slightly deteriorate the flow of reading of sliced content, which makes common sense. However when asking users if this situation had a negative impact upon the readability of content presented to them (Q8 & Q9), no statistical difference could be noticed between content batches ($p > 0.300$). Hence although, a



Q7 I felt that some pronouns within the text were referring to earlier subjects mentioned in paragraphs/sentences which were not included within the content presented.

Q8 I felt that missing parts of original paragraphs/sentences, had a negative impact upon the readability of the content presented to me (in terms of pronouns referring to earlier subjects mentioned in omitted paragraphs/sentences).

Q9 I felt that although parts of original paragraphs/sentences were missing, the content presented was easily readable. (with respect to pronouns referring to earlier subjects mentioned in omitted paragraphs/sentences).

Figure 5.20: Reading Flow

slight decrease in the reading flow of content sliced using a content/AHS agnostic approach could be noticed, the resulting impact upon the overall readability of content delivered was marginal.

5.4.3.2 H2: Content/AHS Agnostic Slicing Performance

Throughout the results presented previously in section 5.4.3.1 and section 5.4.3.1, slices produced by the content/AHS agnostic slicer (CAB, AAF, ATF) consistently depicted lower performances in comparison to their rule based content-specific (CSB) or manual equivalent (MAF, MTF). However, in most cases, differences measured between slices produced by the two slicers were statistically insignificant. Two exceptions to this rule occurred. The first occurred with respect to the ability of the content/AHS agnostic slicer to remove superfluous content from native resources, while the second occurred when analysing any decrease in readability. Albeit, both cases could have been anticipated (since this slicer operates without any prior knowledge of the targeted resources to process), differences measured between slices produced using both approaches were very small (about 10% difference in each case). Moreover, when measuring the impact both cases had upon user experience, no significant differences could be noticed. Notwithstanding these results are promising, it is worth mentioning at this stage that although the task, performed by contestants, was selected among other criteria based upon it's linguistic dependency (section 5.4.2.4), the low impact measured due to readability decrease could be due to the task chosen for this evaluation. Additional investigation should confirm these results within other reuse scenarios (section 6.3).

5.4.3.3 H3: Content/AHS Agnostic Slicing Suitability and Production Costs

As discussed in section 3.3.2, in order to support a large diversity of content requirements, the scalable production of content to AHSs should guarantee the provision of i) large volumes of content, ii) at low production costs, iii) suitable for arbitrary activities performed. Although the first condition is necessarily achieved through the selection of an open corpus reuse strategy, the performance of a slicer with respect to the two other conditions is yet to be examined. For this reason, this section focuses upon investigating the suitability and cost of content produced automatically by a slicing system with respect to content manually produced for

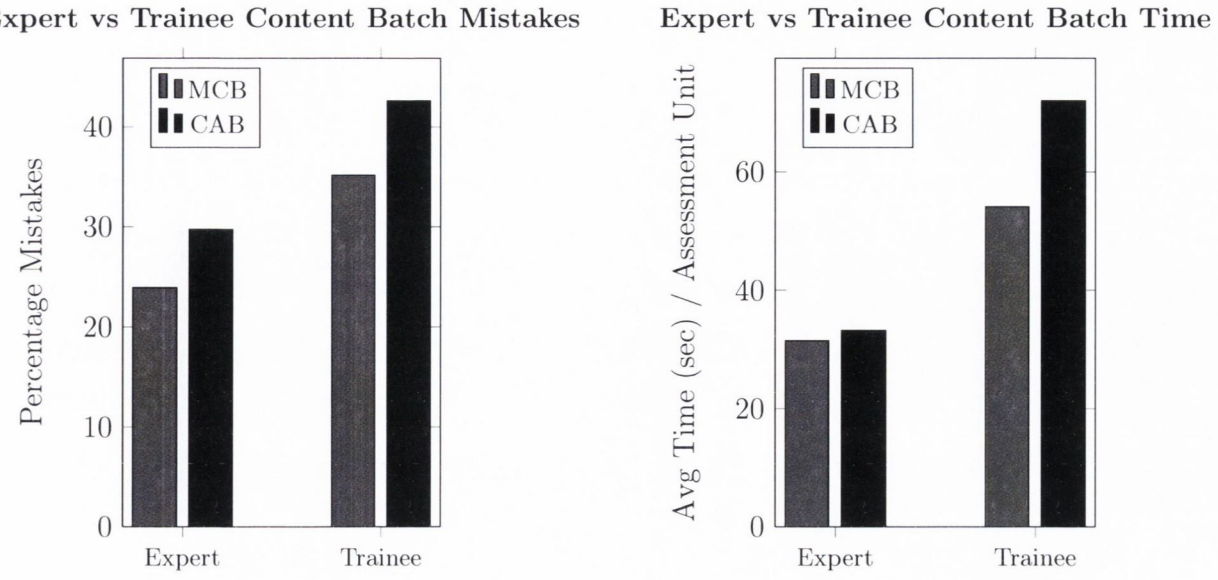


Figure 5.21: Manual vs Content/AHS Agnostic Results

the same activity. The assumption is that, within the context of a specific reuse scenario, if the suitability and costs of content produced by such a technique can be verified for a sample of independently selected niche requirements and open corpus resources, such a technique should scale for any niche content requirements using any open corpus content. Issues detected at this early stage would only be amplified within a large scale deployment making any scaling pointless. The suitability of resources provided by the content/AHS agnostic slicer CAB, with respect to the task performed by contestants, along with the production costs of such content was therefore compared to content manually produced MCB specifically for this task.

With respect to ***content suitability***, results measured for tasks performed by users over both MCB and CAB content batches are presented in figure 5.21. As can be observed, the number of mistakes and time taken to perform e-assessments by the expert group, upon content created using a content/AHS agnostic slicing, appears to be higher than for the content produced manually. Although automated verb chunk annotation recall accuracies measured outperformed those produced manually (Recall A=0.92, M=0.86) (section 5.4.3.1), manual annotations precision accuracies were slightly higher than those produced by the content/AHS agnostic

slicer (Precision M=0.89, A=0.86), which could explain this slight increase in errors. When analysing the results obtained for the trainee group, the same pattern is observable, which would suggest content produced using a content/AHS agnostic slicing approach did induce users in error to some extent. Although the difference in errors between content batches was slightly higher for the trainee in comparison to experts group, an independent t-test indicated this difference was insignificant (mean dif. $E = 5.80\%$, $T = 7.48\%$, $p = 0.8916$), which would suggest that although content generated using a content/AHS agnostic approach did induce users to answer erroneously some assessment units, users from the expert group didn't appear to use their language skills to compensate content suitability differences between both batches. When trainees were asked whether, for content batch CAB, *the number of erroneous assessment units presented was tolerable*, a mean score of 7 out of ten was measured. When asked whether *Overall, I felt this content was adequate to perform a grammar training exercises* both content achieved very similar scores ($MCB = 8.33$, $CAB = 8.57$, $p = 0.536$) with t-tests again suggesting any difference observed was insignificant. These results indicate that although users achieved slightly lower performances on assessments generated using a content/AHS agnostic slicing approach, this tendency didnt appear to affect trainees more than the experts group of users, nor did it appear to decrease the perceived usefulness of the content for the assessment task performed.

With respect to ***production costs***, since the set of pages used as preliminary resources was purposely manually identified and harvested from the web (section 5.4.2.5), no automated open-corpus identification was performed by the slicer during this experiment. Nevertheless, in order to provide a fair comparison with a manual production approach, an estimation of time required by the slicer to perform this task was necessary. Teachers were asked to only identify open-corpus resources matching specific criteria combinations (such as topics covered, tenses etc.). Hence, an automated resource identification time estimation based upon traditional IR services would have created an unfair advantage towards the slicer, since these techniques only provide keyword searches with little guarantee that resources identified do satisfy these criteria. For these reasons, the OCCS focused crawler [Lawless2009]

was considered a fairer option since it provides the means to specify content to be identified based upon a wider range of constraints (including topics covered) and also guarantees resources identified, if any, meet these sets of constraints.

Time measurements obtained for this experiment reveal that when no particular content requirement was specified, teachers took an average of 3.75 minutes to harvest open corpus resources and extract arbitrary fragments suitable for grammar exercises. Requesting resources to be on a specific topic, only slightly increased the average time measured (4min) whereas requesting resources to possess verbs conjugated at particular tenses nearly tripled the time needed (10.5min). These results follow common sense, since the ability of humans to identify topics covered within resources is much more straight forward than for machines, however the reverse is also true when dealing with specific fined grained requirements such as verb tenses. Teachers in average took 4.25 minutes to annotate fragments (189 words in average, 14 annotations per fragments) leading to a total time ranging from 8 min to 14.75 min to produce these resources. The results of this process would require the equivalent of between 1.5 to 2.8 years of manual labor necessary to produce a hundred thousand of such resources. According to [Lawless2009], a time performance of 149,993 valid resources identified in 43h was measured for the OCCS (without any Central Processing Unit (CPU) parallelisation). This is equivalent to $17.2 * 10^{-3}$ minutes of identification time necessary per page. Summing extraction, annotation and slice creation time performed on a 2.8GHz machine leads to a total of $5.4 * 10^{-1}$ minutes necessary to produce each page, which represents a hundred thousand pages produced in 37 days. Assuming no parallelisation was used during the slicing process (section 3), this already represents a difference of up to 96% production time increase with respect to its manual production equivalent. Although automated and manual production time are clearly not directly comparable, one can assume in most cases, server costs per time unit to be much lower than labor costs. Considering how low server production time were measured in comparison to a manual activity, automated content production cost can be inferred to be also much lower than a manual approach and hence more scalable for large number of resources produced.

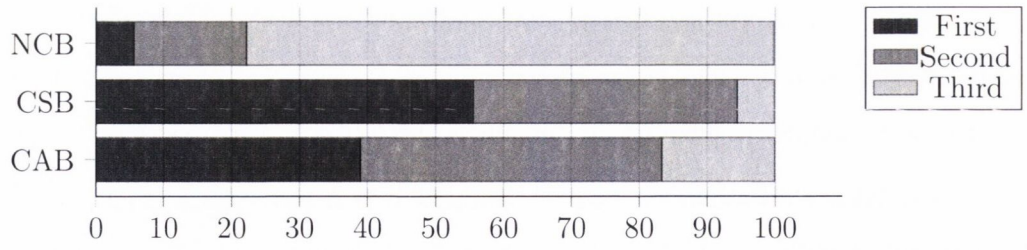


Figure 5.22: Content Batch Ordering

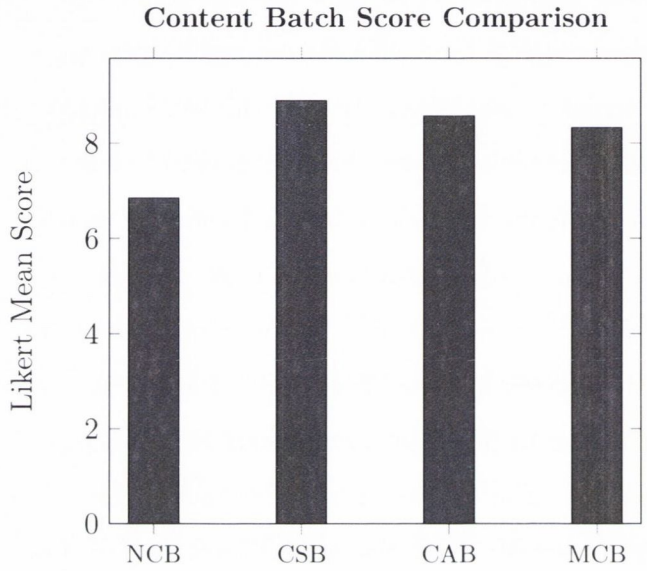


Figure 5.23: Content Batch Preferences

5.4.3.4 Overall Content Satisfaction

Finally, after being presented each content, users were asked whether if *Overall, [they] felt this content was adequate to perform such an task.* Content-specific (CSB) and content/AHS agnostic (CAB) batches achieved the best scores, with means respectively equal to 8.89 and 8.57 in comparison to native content (NCB), which only achieved 6.85 ($p < 0.034$) (figure 5.23). Surprisingly, content batch MCB only achieved a score of 8.33. The difference with both content/AHS specific and agnostic batches however was insignificant.

Moreover, when asked to order contents in order of preference, 56% of users placed CSB content in first position (figure 5.22)[14], closely followed by CAB, with the overall

[14]Due to a data gathering error discovered after the experiment was carried out, the data collected

majority placing content in its native form (NCB) in the last position. These overall results confirm and reinforce the trend observed across variables, which suggest the reuse of existing resources was improved through the approach of slicing, with content/AHS agnostic slicing achieving very close performance to its content-specific equivalent.

for batch MCB was corrupted. For this reason, this batch is omitted from this graph

5.4.4 Result Summary and Discussion

This section presented an experiment, applied within the application domain of a language e-assessment scenario, and designed to evaluate and compare alternative forms of open-corpus reuse. More specifically, the reuse performance and quality of the content delivered by the slicer, implemented for the purpose of this research, was evaluated via a user trial.

Results of this experiment provide evidence that the process of slicing open-corpus resources does improve their reuse, while at the same time minimising any decrease in the quality of original content delivered (H1). In particular, the ability to remove superfluous content from these resources appeared to have a significant impact upon their reuse. Despite depicting lower performances, results obtained for content produced using a content/AHS agnostic approach to slicing were very close to its content specific equivalent (H2). Differences were all statistically insignificant with exceptions occurring with respect to the correct removal of superfluous content as well as decreases in reading quality of content produced. Both results depicted a decrease in performance of around 10% for content/AHS agnostic batches. In both cases however, the impact perceived by users was insignificant. Although tasks accomplished by users were selected specifically based upon their dependency over content quality (section 5.4.2.4), additional experiments investigating the impact caused by such a decreases in performance, within alternative use cases, would nevertheless be beneficial.

Similar results were obtained when comparing suitability performance between open corpus content reused manually with content reused using a content/AHS agnostic slicing approach (H3). Results indicate that although content/AHS agnostic slicing constantly portrayed slightly lower performances than those manually produced, these differences had no noticeable impact from the point of view of users. Nevertheless, when taking into account production costs of both content batches, resources generated using a content/AHS agnostic approach significantly outweighed those manually produced. Hence, in the context of a high speed, low cost production environment, one could easily assume any content produced with unsatisfactory

suitability to be discarded and rapidly replaced, which could compensate any decrease in suitability.

Results obtained as part of this evaluation appear to indicate that a content/AHS agnostic slicer can improve the reuse of open corpus resources within AHSs through its ability to deliver these resources i) as right-fitted slices, ii) without any pre-determined conditions[15] and iii) based upon a variety of AHS niche content requirement specifications, undetermined in advance. Content/AHS agnostic slicing reuse of open corpus resources would thus appear to represent a promising approach to consider for the delivery of resources within AHSs. Results indicate that its ability to produce large volumes of content on-demand, at very low costs and with a suitability comparable to manually produced resources, could provide a scalable solution to the content delivery needs of AHSs according to requirements presented in section 3.3.2.

It may be argued that the reuse scenario (including the choice of what constitutes discerning meta-data) selected for the purpose of this experiment only represents one specific use case of open corpus reuse. Section 5.5 of this thesis, for example presents the same slicer implementation used to serve other use case scenarios. However, further investigations will be required to determine the full range of reuse scenarios possible (section 6.3), which slicing approaches could support. However, this experiment strongly suggests that the ability to go beyond the one-size-fits-all delivery of open corpus content and automate the right fitting of such resources for reuse within various content consumers is possible, with a minimal decrease in quality of this original content.

[15]Within the range of content considered for this research (section 5.2.10)

5.5 Independent AHS Evaluation

5.5.1 Evaluation Objectives and Hypothesis

The results obtained in the experiment presented in section 5.4, suggested that the use of content/AHS agnostic slicing techniques does improve the reuse of open corpus resources, with a performance close to content-specific and manually produced resources, at much lower costs.

The reuse vehicle built and selected for the purpose of this experiment however did not technically represent a pure content/AHS agnostic slicer/slice consumer relationship. Design requirement DR 2 (section 3.3.2) defined a content/AHS agnostic slicer as a system which produces slices i) for an non-established set of AHSs, built independently of the slicer, using techniques which do not require any predefined knowledge of either ii) the resources targeted or iii) the reuse purposes (i.e expressed as niche content requirements). Although the open corpus resources (ii), as well as the combination of niche content requirements (iii), provided as an input to Slicepedia were indeed unknown to its designers, both the Slicegap slice consumer and language use-case scenario were known in advance by Slicepedia designers. For this reason, the previous slicer/slice consumer relationship comes short of correctly fulfilling requirements (i) and (iii).

The experiment presented in this section hence investigates whether a set of open corpus resources, unknown at design time (ii), reused via content/AHS agnostic slicing techniques, could be recomposed within independent AHSs (i) in order to serve use-case scenarios unknown in advance by slicer designers (iii).

Hypothesis:

H1) Content produced by a slicer can be successfully recomposed within an independent AHS system

5.5.2 Slice Consumer Selection

While the slice consumer selected for the previous experiment aimed at minimising interface complexity, in order to assess resources individually using a simple interface, the slice consumers presented in this section in contrast provide an independent user interface, independently recomposing slices with arbitrary proprietary content.

The slicer prototype, implemented for the purpose of this research, was therefore integrated as a content delivery service within an independent eLearning AHSs. The chosen AHS combined proprietary manually produced content, specifically for the purpose of each platform, along with slices produced automatically and on-demand by Slicepedia. Hence, in comparison with the experiment carried out in section 5.4, results for this experiment were measured through an independent AHS user interface, built without any Slicepedia designer involvement. In other words, decisions regarding how slices were delivered to each AHS would be selected, recomposed or reused were solely taken by the team responsible for each AHS.

The AMAS AHS consisted of an upgraded version of the Adaptive Media and Services for Dynamic Personalisation and Contextualisation (AMAS) AHS prototype [Staikopoulos2012], targeting undergraduate university students in the area of Structured Query Language (SQL) programming (figure 5.25). The platform was built independently of Slicepedia (i), for a use case scenario unknown in advance by Slicepedia designers (iii).

The underlying architecture of the AMAS and PMCC platforms are very representative of most modern AHSs (section 2.2.2). For this reason, these platforms appeared to provide a valid representative example of hypothetical contemporary slice consumers. Both AHSs are built through the combination of standard modules consisting of a narrative component, a user model, a content model, an adaptation engine as well as a portal displaying the resulting personalised experience to users. In addition to a content model, the AMAS platform also contains a service model enabling it to adapt various services based upon user needs.

Figure 5.24 presents the architecture of the AMAS AHS [Staikopoulos2012,

Staikopoulos2013]. As in most adaptive courses, the platform initially requires the educator designing a course to input a set of three fundamental building blocks. These consist of i) a list of concepts and tasks from the learning domain (e.g. SQL commands and triggers) as well as ii) sets of learning activities based upon these concepts or tasks (e.g. Building an SQL table etc.). Finally, a iii) set of adaptation rules that govern how the concepts and tasks should be adapted based upon various contexts (e.g. for novel/expert learners) is also required. All of these elements can be produced using an authoring tool such as Generic Responsive Adaptive Personalized Learning Environment (GRAPPLE) [Conlan2013F]. These three elements are then combined by AMAS into a *narrative*, which can be evaluated and processed by the *adaptation and personalisation engine* module (S). The latter analyses the narrative it receives and produces from it a *personalised learning activity* for each user by taking into account their individual characteristics stored within the *user model* (W). This personalised learning activity describes the learning workflow (concepts and tasks), which will be presented to each individual user. It is then handed over to the enactment engine (T), which is responsible for gathering all the components necessary to build this specific learning activity. For each concept or task encountered within the personalised learning activity, the enactment engine requests sets of content and/or services to the match making module (U), which attempts to return the most adequate content or service, based upon the combination of prerequisites related to each concept and tasks it is provided with. This is performed by analysing in detail the meta-data contained within the *content* (Y) and *service* (X) models. URIs pointing to the most adequate content and/or services identified are then returned to the enactment engine which forwards them to the *portal*. The later finally presents the learning activity to each user by combining each content and/or services, referred to by the enactment engine, into a coherent presentation.

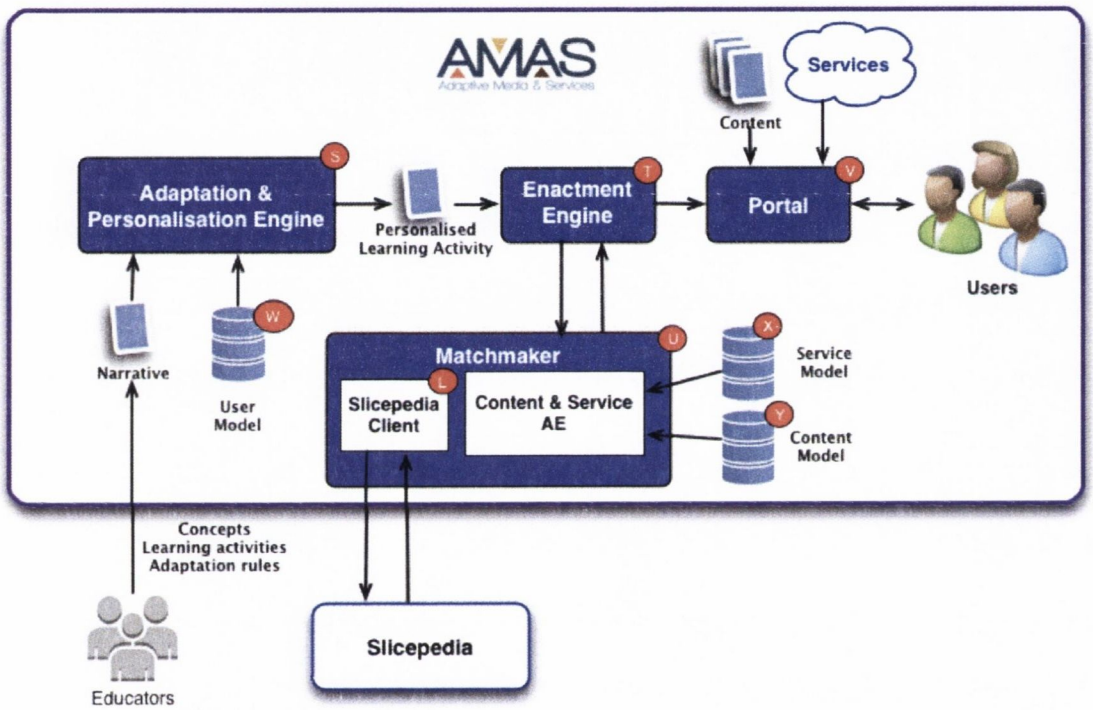


Figure 5.24: AMAS Architecture

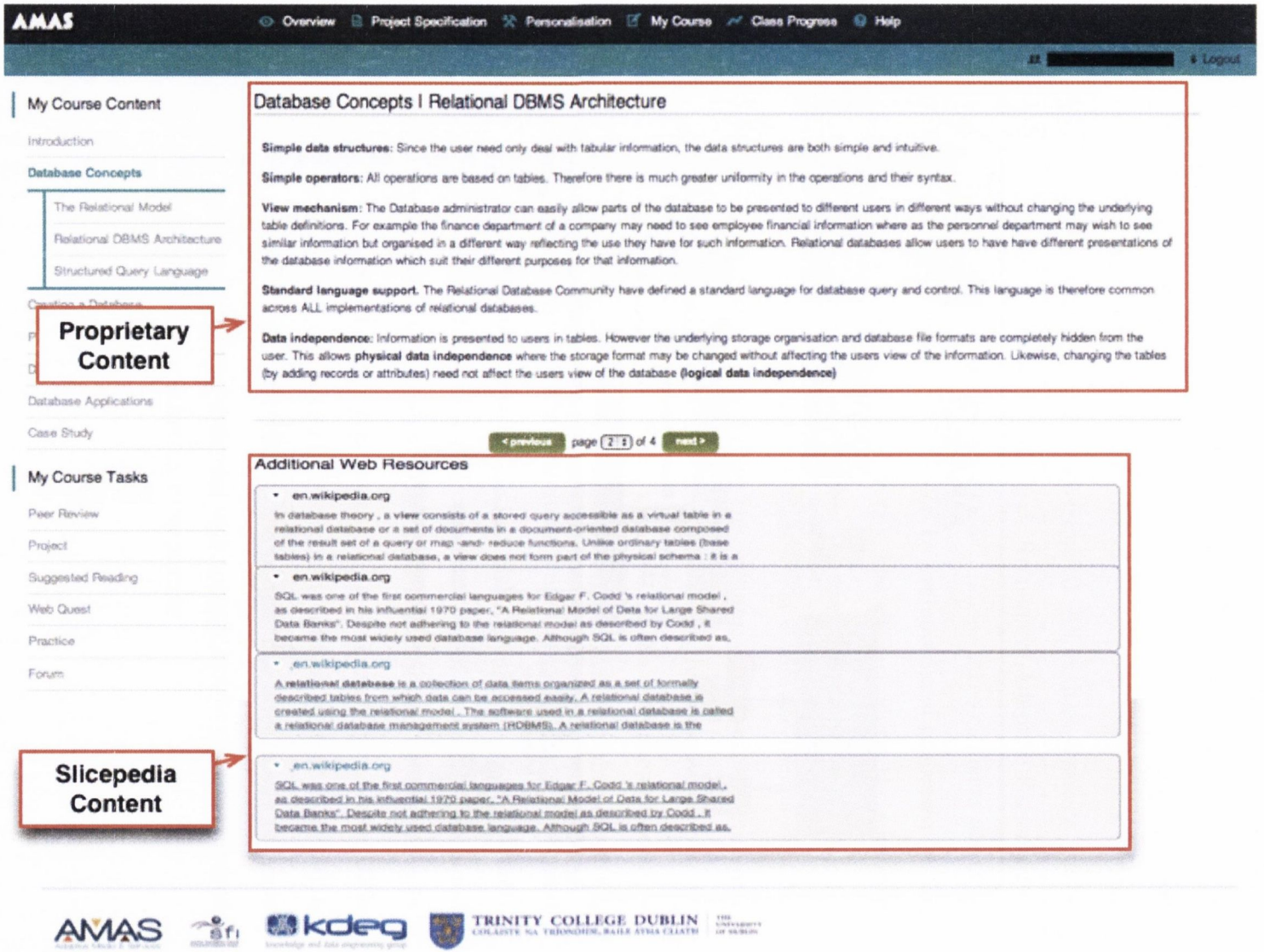


Figure 5.25: AMAS Interface

5.5.3 Content Production

In addition to the proprietary resources produced in advance by educators (selected by the AMAS team), Slicepedia content was incorporated within the AMAS AHS. As is described in figure 5.24, the Slicepedia service was integrated within the AMAS platform through the matchmaking component. For each concept encountered within the personalised learning activity, the enactment engine requested both proprietary and open content. Proprietary content was identified by the matchmaker by analysing meta-data contained within the content model as described above, whereas open content on the other hand was identified by querying Slicepedia (using keywords and/or conceptual searches depending on DBpedia concepts tagged) directly through the use of a Slicepedia Client library, provided by Slicepedia designers (section 4.5).

While proprietary content, related to each topic covered by the course, was accessed as pre-existing flat files through a local repository, Slicepedia content on the other hand was produced and delivered to AMAS in real time. The slices were produced by Slicepedia on-demand, based upon niche content requirements submitted by the AHS with respect to individual user interaction. In other words, slices were produced only after the selection by individual users of arbitrary concepts. Hence, although prior open corpus harvesting was performed by Slicepedia prior to any user interaction (see below), the delivery by the AMAS AHS of niche content requirements to Slicepedia was performed only as users were interacting with the platform.

As described above, since the AMAS system separates the meta-data related to proprietary content from the content itself, only URI references pointing to this content are returned by the matchmaker. With respect to open content on the other hand, the content itself is returned directly to the enactment engine. Prior to doing so, the slices received by the matchmaker were also processed in order to extract only the content to be presented to users (i.e metadata contained within the slices was not displayed to users).

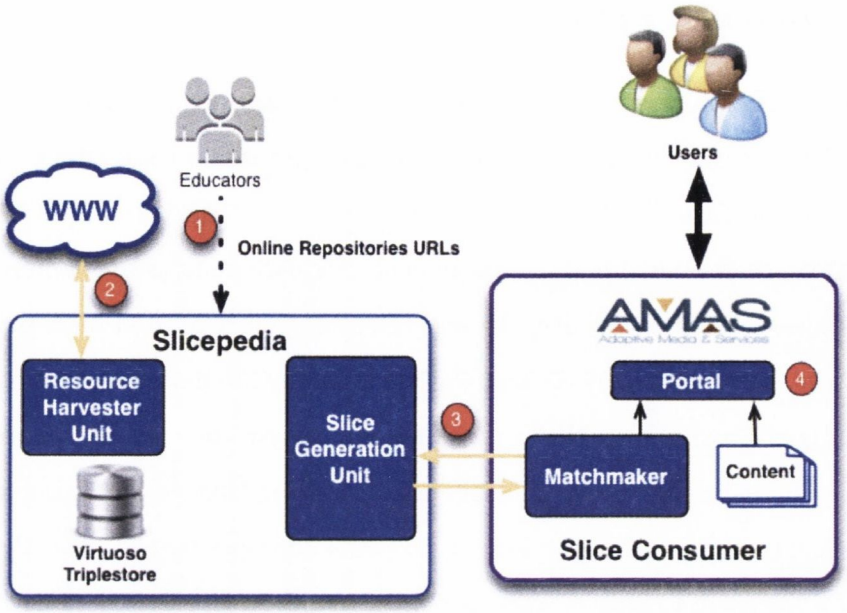


Figure 5.26: AMAS Content Batch Creation

This integration between the two platforms was put in place by way of the following four steps (figure 5.26).

Step 1/ Open-corpus Repository Specification: AMAS educators were initially requested to provide an arbitrary list of online repositories containing resources they wished to target for slicing. The list provided to Slicepedia ultimately consisted of six websites, mostly consisting of online SQL tutorials.

Step 2/ Slicepedia slicing: Repositories provided were then crawled[16] by Slicepedia, harvested and prepared for reuse according to a Slicepedia request of type SR_1 (section 4.2.2).

Step 3/ Slice delivery: Finally, once all pages were processed, slices where generated (using Slicepedia requests of type SR_3) and delivered to AMAS in real-time based upon predefined SQL topics contained within the course and selection choices made by users.

Step 4/ Slice recomposition: Groups of slices received per topic were finally recomposed by the AMAS system into an accordion type interface (see figure 5.25).

[16]The 80Legs crawler was set with a depth crawl equal to two

5.5.4 Experiment Scenario

For this experiment a group of 88 undergraduate students, attending a third and fourth year university database course, were presented during one semester with an online personalised SQL course consisting in the AMAS AHS platform presented above. This authentic case study was overall very similar to a previous study carried out by [Staikopoulos2012]. As part of this course, students were required to complete various database related tasks (such as designing and implementing a database). For each activity, students were presented with SQL related proprietary material covering topics required to be mastered, which they could study in order to perform these tasks. In contrast with the original use-case presented by Staikopoulos however, the platform also incorporated content dynamically generated by Slicepedia on demand for each arbitrary topic and recomposed by AMAS as seen in figure 5.25. This experiment was run by the designers of the AMAS system independently from Slicepedia designers.

Following the completion of all tasks, users were presented with a questionnaire answered using a five point Likert scale. In addition to existing questions related with the AMAS system itself, the designers of this AHS kindly agreed to include additional questions related to Slicepedia content. Hence, for continuity purposes, questions related to the content delivered by Slicepedia to AMAS were similar in nature than those presented to users in section 5.4.

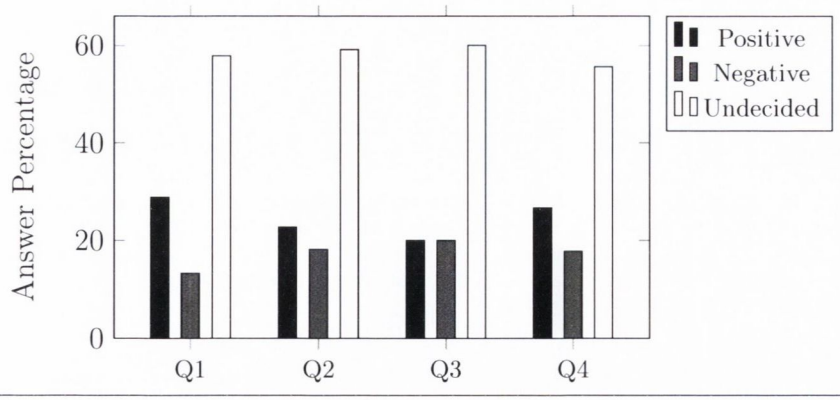
5.5.5 Results

Among the group of 88 students which accomplished all the SQL tasks presented to them, 50 students provided answers to the questionnaire at the end of the semester. Figure 5.27 present the results obtained from this questionnaire, with respect to the content produced by a slicer for recomposition within an independent AHS system. As can be seen, results obtained for questions related to the Slicepedia content, recomposed within AMAS, depict a very large proportion of undecided users. This situation occurred as well on many of the other AMAS related questions presented to users.

For the purpose of the previous experiment (presented in section 5.4), a ten point likert scale was chosen in order to push users to make a judgement. The questionnaire presented to undergraduate students for the purpose of this AMAS experiment however, provided instead a five point likert scale[17]. This decision might have induced users not to make any decision and choose a more neutral position overall. Also it transpired that, based upon user comments analysed while processing the results, some students had not noticed the Slicepedia content on the platform at all! It turned out that this situation occurred depending on the screen size used by each student. In some circumstances, the Slicepedia content appeared below the level of the screen and required a lot of scrolling, which might explain why some students didn't notice this content. Finally, a last possible reason which could perhaps explain this large number of undecided users across the entire questionnaire, could be due to the fact that the additional Slicepedia questions were provided to students at the end of all the other AMAS questions. According to the lecturers, present while the questions were being answered by students, the time available to go through each question was relatively short. Hence, the large number of undecided users could be also a result of this combined situation.

Nevertheless, despite the large number of undecided users, the results obtained do provide some indication into how Slicepedia content was perceived overall within the

[17]As the experiment presented in this section was carried out as part of a larger experiment, the decision concerning this scale was taken by the AMAS team



Q1 The Additional Web Resources were of similar quality to the rest of the content presented within the website.

Q2 The Additional Web Resources were missing portions of content, making them hard to understand.

Q3 The Additional Web Resources contained irrelevant material which should have been removed to make them more easily understandable

Q4 The Additional Web Resources were beneficial in learning the topics covered by the course.

Figure 5.27: Slicepedia resources within the AMAS platform

AMAS AHS by students which did interact with it. When asked whether the resources provided by the slicer *were of similar quality to the rest of the content presented* to them (Q1), 29% of contestants provided a positive answer in contrast with only less than half this amount (13%) responding negatively. This is a very positive results, suggesting the quality of resources produced via a content/AHS agnostic slicing could perform reasonably well in comparison to that manually produced in advance for a dedicated platform. On the other hand, when asked whether *missing portions of content [made resources] hard to understand* (Q2), a slightly higher proportion of students agreed with this statement in opposition to those who disagreed (23% and 19% respectively). Although this represents only a 4% difference, this results appears to confirm the reading quality decrease measurements obtained earlier for the experiment carried out in section 5.4 with content/AHS agnostic content. When asked whether these resources *contained irrelevant material* (Q3), an equal amount of

users surprisingly responded both positively and negatively, which does not provide enough conclusive evidence on the slicers performance with respect to this dimension. Finally, when asked whether these resources *were beneficial in learning the topics covered* (Q4), 27% of users responded positively with only 18% disagreeing, which provides some indication that the slices provided by Slicepedia did contribute to the overall experience of users on the AHS platform.

5.5.6 Result Summary and Discussion

Despite a large number of undecided users, these results appear to indicate that the incorporation and recomposition of content/AHS agnostic content within this independent AHS provided a positive experience to users; both in comparison with proprietary content as well as in terms of the ability for such content to support this learning experience. This would suggests a content/AHS agnostic slicer can indeed be built to support the production of content for reuse use-cases unknown in advance by its designers within independent AHSs. The results obtained as part of the AMAS experiment also appear to reinforce previous results obtained in relation to possible decreases in reading quality.

As mentioned in section 5.5.4, as this experiment was conducted in conjunction with an independent evaluation of the AMAS AHS, only a small set of questions could be presented to users, which clearly limited the extent to which the analysis of this integration could be performed. Nevertheless these results, do represent an encouraging first step with respect to the ability to supply slices, produced by a content/AHS agnostic slicer to independent AHSs. Additional experiments, currently underway with the AMAS team, are investigating the usage of Slicepedia's open content within the same scenario, using a different recomposition interface[18] as well as with an additional set of quantitative metrics. Finally, during the interaction with the AMAS team, it transpired that most of the niche content requirement dimensions of interest to this platform evolved mostly around topical and granularity dimensions of the slices produced. Further experiments, investigating the usage of slices within

[18]With the screen sizing interface issue mentioned in section 5.27 solved

a scenario involving a more diverse range of content requirements would be very valuable along with a correlation between the impact of varying content requirement requests upon users.

5.6 Conclusion

This chapter presented a set of evaluations aimed at addressing the third objective of this thesis; namely to evaluate the overall performance of the approach in general as well as the slicer prototype implemented and presented in the previous chapter. Analysis of the fragmentation approach selected for the purpose of this prototype was presented along with user-trial evaluation of the overall prototype implementation. Various trade offs and limitations of both the fragmentation approach as well as the slicer prototype developed for the purpose of this research were discovered. The following chapter presents a summary of what was performed for the purpose of this research along with a summary of overall results obtained from these evaluations.

Chapter 6

Conclusion

6.1 Introduction

This chapter concludes this investigation by providing a summary of what was achieved with respect to the overall goals and objectives enunciated in the first chapter of this thesis. Specifically, the experimental results obtained, as well as the limitations of the prototype and approach proposed by this research are discussed. Additionally, the overall contribution of this research and potential directions for future research are also outlined. This chapter begins in section 6.1.1 by reiterating the research question addressed within this thesis, as well as each underlying objective. What was accomplished for each of these objectives is summarised and discussed. Section 6.2 thereafter discusses the overall contributions of this investigation including research publications that have resulted from this work. Finally, section 6.3 concludes this chapter by presenting areas which will require further investigations to be carried out, extending the work presented in this thesis.

6.1.1 Research Question

As stated in chapter 1, the investigation carried out within this thesis is "focused upon the design and development of a service that improves the reuse of open corpus content available on the WWW (section 1.2). More specifically, the research question investigated in this thesis is: What are the techniques and technologies required to improve the reuse of open corpus resources within independent AHSs? By reuse improvement, this thesis focused upon the ability of AHSs to request open-corpus resources *i*) without any pre-determined conditions *ii*) as right-fitted content packages, *iii*) based upon a *variety* of possible AHS content requirement specifications, undetermined in advance."

In order to address this question, the thesis has proposed the notion of a slicing approach to open corpus reuse. This approach enables the harvesting, right-fitting and delivery of open web content as slices, which can be directly incorporated and recomposed within independent AHSs.

This approach has been achieved through a novel combination of techniques from the areas of NLP, IR and semantic web. These techniques have been integrated and evaluated within a slicer called Slicepedia. The results of this investigation indicate that such an approach *i*) improves the reuse of open-corpus resources (through its ability to right-fit content on demand, for a variety of content requirements), *ii*) without the need for any pre-determined conditions concerning open-corpus content targeted for reuse. This approach *iii*) could produce resources recomposable and reusable within independent AHSs, *iv*) for use case scenarios unknown in advance to slicer designers, and *v*) could support the scalable supply of resources for platforms with a large number of individual user needs (by delivering resources of similar quality than that manually produced for the same activity at much lower costs)

As mentioned in chapter 1, the underlying research process chosen for this investigation involved focusing upon the overall end-to-end reuse of open corpus resources (from harvesting to reuse within independent AHSs) as opposed to individual steps of this reuse. For this reason, although the experiments carried out

were achieved over the widest range of resources possible (based upon the ability of state of the art components used), this range was narrowed down to a subset of resources (news or encyclopedia articles) due to the limitations of the fragmenter component of the pipeline selected for this research. Nevertheless, this range is sufficiently large and diverse to prevent the slicer in determining in advance the resources being selected for reuse and therefore it is still a satisfactory range for the purpose of the argument supporting content agnosticity capabilities of such reuse.

Given that the result of this research indicates content agnostic slicing is achievable with a performance close to content specific slicing, future research should focus upon expanding the range of open corpus resources targetable by an agnostic approach. In the event targeting a larger range isn't feasible, several content-specific slicers could be used instead. However, as mentioned in chapter 1, doing so will also increase maintainability requirements. For this reason, when possible a content agnostic slicer should be preferred in comparison to a content specific approach.

Secondly, as evaluating the recomposability of slices within an array of most popular existing independent AHSs was impractical, the independent AHS chosen for the case studies carried out as part of this research was chosen based upon its architecture being typical of modern AHSs. The decision to choose an AHS with a similar architecture to most state of the art system was taken to alleviate as much as possible this impracticality. The author of this research is fully aware however that additional experiments, using various AHSs, will be needed to confirm this claim.

A series of trials, involving authentic real life user needs, was conducted, which suggests that the use of a slicer can significantly improve the reuse of open corpus resources, while at the same time minimising any decrease in quality of the original content. The suitability of resources presented to users also appeared to perform very well within user trials in comparison with content specifically produced for each sets of tasks.

In addition to addressing the overall research question presented in chapter 1, this thesis, along with publications resulting from this research[1], has also met the

[1]The publications contributing to each objective are listed in the contributions section

individual research objectives identified in (section 6.1.1). Each of these objectives is discussed individually below.

6.1.2 Research Objectives

Research Objective (1): *To investigate and specify techniques and technologies which can be used to enable automated reuse improvement of open corpus resources capable of supporting niche content requirements of independent AHSs.*

In order to achieve this objective, an integral part of this research involved conducting a state of the art review of content reuse, adaptation and supply approaches in use in the research community in general, as well as in AHSs.

This review identified and classified various content reuse paradigms, used extensively within the research community in general. The review subsequently presented the overall architecture of modern AHSs and more specifically, focused upon the interaction of content models in such systems with other parts of their architectures. This was later followed on by an analysis of various AHSs content supply approaches available in the literature and their limitations. This initial review led to the identification of limitations in the content supply approaches currently in use, along with potential opportunities for improvements.

The second part of this review presented a review of content reuse and analysis approaches. More specifically, an overview of successful content reuse approaches, as well as a detailed review of existing content fragmentation algorithms was carried out, since this process was identified at an early stage of this research as representing a critical component of any automated open corpus reuse. This content fragmentation review also served as a basis for the selection of an algorithm to be used within the content supply prototype implemented for this research. Finally, a review of state of the art NLP algorithms used for content analysis purposes was also carried out.

Research Objective (2): *To design and implement a content/AHS agnostic slicing service, based upon these techniques, which can provide the on-demand harvesting, customisation and delivery of open corpus content to AHSs.*

Motivated by the analysis presented in chapter 2 of existing limitations in AHS content supply mechanisms, as well as advances in the state of the art of NLP, IR and semantic web techniques, an approach to AHS content provision through slicing was proposed, which enables the customised reuse of open corpus resources within independent AHSs.

The slicer architecture proposed in chapter 3, addresses all aspects of this research objective, as it enables the generation of content packages (consisting of slices) from open-corpus resources, right-fitted according to a large range of possible content requirements (defined through the slicers CAS), without the need for any pre-determined conditions concerning open-corpus resources targeted for reuse (through the selection of internal components operating independently of content processed). Since the entire architecture is fully-automated, the slicer architecture proposed can support the on-demand provision of content requirements (by AHSs to the slicer), as well as the on demand harvesting, customisation and delivery of open corpus resources. In order to support the possibility of providing larger volumes of resources, as well as wider content requirements specifications, the slicer architecture proposed for this research was also designed so as to enable the creation of a so-called *web of slices*, achieved through third party institution collaboration.

In order to evaluate this approach, a prototype implementation of this architecture, called Slicepedia, was developed and applied within different authentic use case scenarios. As an initial step, this content/AHS agnostic implementation focused on a subset of HTML based open corpus resources, available on the WWW, written in the English language. Nevertheless, in order to permit future multilingual slicing investigations to be carried out using the same architecture, whenever English specific components were selected for this implementation, references to alternative multilingual components available to the community were provided in chapter 4. In order to encourage the adoption of slicing practices by the community, for the purpose of third party institution collaboration, the implementation carried out for this research was also achieved using only free open source or SaaS components.

Evaluations of the Slicepedia prototype have shown that this research objective

has been met successfully, as the slicer was able to receive combinations of content requirements from AHSs, unknown in advance by Slicepedia designers, and produce slices on demand (through the harvesting and right-fitting of open corpus resources) subsequently delivered and reused for various AHS specific tasks.

The design and implementation of Slicepedia is documented in chapter 3 and chapter 4 and a summary of how each design requirement was supported is provided in section 5.3.

Research Objective (3): *To evaluate the extent to which such a service can improve the reuse of open corpus resources within independent AHSs.*

This objective has been achieved through a series of evaluations investigating the performance of individual components of the slicer, as well as alternative prototype slicer implementations within different AHSs. Most of these experiments were each performed over parallel sets of content simultaneously. More specifically, performance was measured, within authentic user-trial scenarios, in terms of fragmentation accuracy, suitability of content to perform tasks, right-fitting accuracy and production costs of content produced. A multitude of metrics have been used throughout these experiments, including ARI, sizing percentage index, time performance (for fragmentation accuracy and production costs), as well as task completion time, number of mistakes, perceived suitability, annotation precision and recall, content summarisation quality (for content suitability and right-fitting performance).

Since the process of structurally fragmenting resources represented a critical component the slicer architecture presented within this research, the first evaluation carried out for this investigation consisted of an analysis of DCF algorithms selected, based upon the state of the art review, as the most promising approach at the time. The analysis confirmed that this approach to content fragmentation could indeed process any HTML resource individually, regardless of tag structure, domain or language used. However, limitations related to time performance, parallel multilingual accuracy and content type dependencies were also discovered. Since multilingual slicing was considered out of scope of this research, this aspect of the research was postponed for future work. Time performance limitations however were addressed

by proposing a new variation of DCF algorithms, which stabilises time performance across fragment output sizes. Limitations due to content type dependent accuracies on the other hand, required subsequent experiments to narrow down the range of resources targetable by the slicer to a subset of WWW content. Since this subset represents a very large volume of open corpus resources, this limitation didn't prevent the rest of this research, related to open corpus slicing, to proceed. Nevertheless, as mentioned in section 6.3, this discovery represent an important limitation, which should be addressed if such a fragmentation approach is to be selected permanently for content slicing purposes.

Following this analysis, and with the intention of validating the prototype implementation developed for the purpose of this research, a summary detailing how each implementation decision taken within the previous chapter contributed to design requirements (enunciated in chapter 3) was presented. Since many implementation decisions contributed to several design requirements simultaneously, a graphic summary of these requirement implementations dependencies was also provided to the reader for clarification purposes.

Following this implementation validation, the entire slicer prototype was then evaluated with respect to its ability to i) improve the reuse of open-corpus resources within AHSs (through right-fitting) and ii) provide content suitable for an arbitrary task iii) for platforms requiring large diversities of unpredictable content requirements. The results of this evaluation have shown that the process of slicing open corpus resources does improve its reuse within AHSs and can be achieved using content/AHS agnostic techniques with minimal decrease in performance relative to content-specific techniques. Decreases in performance of about 10% were measured with respect to the successful removal of superfluous content within native resources, as well as with the ability of the slicer to preserve the original reading quality of native resources. Although users did notice these differences, any impact upon the tasks performed were measured as statistically insignificant in both cases. With respect to the suitability of resources delivered by Slicepedia, results show that content automatically produced by the slicer constantly performed lower in comparison to

content manually produced for the same activity. Although content produced by the slicer did appear to induce trial participants in making slightly more mistakes, users however perceived this difference as being tolerable. Moreover, although the process of slicing open corpus resources did affect the reading quality of content reused, users with higher levels of English did not appear to use their language level to compensate differences in content quality. Results for this experiment also indicate that, despite quality decreases with respect to content produced, the benefits of using a slicer for the purpose of AHS content provision outweigh those offered by the manual production of similar resources for very large number of diverse users. While the latter can provide highly curated quality content, the costs involved in the production of such resources prevents such a service to be deployed to large number of users with diverse needs. The ability of a slicer, on the other hand, to produce resources with a quality close to that manually produced, at much lower costs and on-demand based upon user needs, would represent a much better alternative overall, without the need to predict in advance user needs. In such a scenario, content with insufficient quality could easily be discarded and replaced.

In the fourth evaluation presented in this thesis, the slicer prototype was used as a content provision service, integrated within two individual AHSs built independently of the slicer, for use case scenarios unknown in advance to slicer designers. Content delivered by the slicer was recomposed within this AHS along with proprietary content produced specifically for that AHS. Overall, user perception of content delivered by the slicer performed well compared to the proprietary content, and users perceived this content to be beneficial to the task performed. In this experiment, reading quality performance of content supplied by the slicer also depicted lower results than proprietary content which confirms previous results.

Overall, the evaluations carried out within this experiment involved more than 120 anonymous participants in online based user studies. These experiments suggest that the reuse of open corpus resources within independent AHSs can be improved by a slicing approach. They also suggest this approach can be achieved through the use of content/AHS agnostic techniques with similar performances to manual

or content-specific alternatives at the price of slight decreases in content quality. Moreover, the ability of content/AHS agnostic slicing to reuse open-corpus resources on-demand, without any pre-conditions, in order to meet niche content requirements unknown in advance of content requests, provides a significant advantage over state-of-the-art alternatives when targeting independent AHSs involving large number of users with wide and unpredictable diverse needs.

6.2 Contributions

The major contribution of this thesis is a novel approach to ***content/AHS agnostic open corpus content reuse within AHSs***, which combines NLP, IR and semantic web techniques. In particular, this innovative approach is capable of harvesting open corpus resources and customising them, on demand, according to specific niche content requirements provided by individual AHSs, fully automatically. Moreover, this approach enables a better integration of open corpus resources within various independent AHSs, via a new content supply service loosely coupled with individual AHSs. The separation of concerns between the supply of content and its composition provided by this service, enables AHS designers to focus on the adaptation and composition of resources presented to users rather than the provision of such resources. Several user-trials conducted within this investigation, suggest this approach improves the reuse of open corpus resources within independent AHSs, with minimal decrease in quality of original resources targeted for reuse, at low costs. This approach thereby advances the field of adaptive hypermedia by providing a novel, independent content supply service, providing large volumes of customised resources to AHSs, on demand.

In addition, the thesis has presented an ***architecture called Slicepedia***, which implements this approach by combining the harvesting of open corpus resources, NLP based content adaptation and semantic web data representation techniques. This architecture has been used as the basis for a series of prototypes, which have demonstrated the successful application of this content supply approach in authentic eLearning tasks. The prototype service has been shown to successfully harvest, customise and deliver open-corpus resources as content packages, on demand, in order to meet niche content requirement needs of independent AHSs. The benefits obtained by using an automated and customised open corpus reuse approach, has uncovered new directions for the development of novel AHS content supply services as well as novel open and user-driven AHSs, availing of new opportunities offered by such content services (see section 6.3). In particular, it has been shown that open-corpus content reuse prototypes can be built using techniques which do not

require the need for any pre-conditions of resources targeted for reuse. Moreover, it has been shown that a service using such techniques can successfully support specific content needs of individual AHSs without any prior knowledge of such requirements.

The contributions of this research have also resulted in a number of high-quality scientific publications as well as a patent filed at the US patent office. In addition to the scientific contributions of this thesis, the presented research work has also received commercialisation funding from the Science Foundation Ireland (SFI), which provides additional resources to the research and development of the presented techniques and technologies. This funding will be dedicated to pursuing future work proposed in section 6.3.5, and build a next generation slicer-based, user-driven and open, language learning AHS commercial prototype, within the lines of the initial hypothetical use-case scenario presented in the design chapter of this thesis (section 3.3.2.1). Those contributions are listed below.

Patent and Commercialisation Fund Awarded:

- **Patent Filed**: Levacher, K., Lawless S., O'Connor A., Wade V. (2011) "A network system for generating application specific hypermedia content from multiple sources", *US Patent Office* (contribution to objective 2)
- **SFI TIDA Award**: Technology and Innovation Development Award Feasibility 2012 *"Linguabox: Automated Open Content Repurposing Service to support Personalized eLearning"* (SFI 12/TIDAI2441)

Academic Publications:

- **Journal:** (Accepted for Publication) Levacher, K., Lawless S., Wade V. (2013) "Slicepedia: Content-Agnostic Slicing Resource Production for Adaptive Hypermedia" *In the proceedings of the international Journal of Computer Science and Information Systems (ComSIS'13)* (contribution to objective 2 & 3)
- Levacher, K., Lawless S., Wade V. (2012) "Slicepedia : Towards Long Tail Resource Production through Open Corpus Reuse" *In the proceedings of the 11th international conference on Web-based Learning (ICWL'12)* (contribution to objective 2 & 3)
- Levacher, K., Lawless S., Wade V. (2012) "Slicepedia: Automating the Production of Educational Resources from Open Corpus Content" *In the proceedings of the European conference on Technology Enhanced Learning (EC-TEL'12)* (contribution to objective 2 & 3)

- Levacher, K., Lawless S., Wade V. (2012) "Slicepedia: Providing Customised Reuse of Open-Web Resources for Adaptive Hypermedia" *In the proceedings of the 23rd international conference in Hypertext and Hypermedia (HT'12)* (contribution to objective 2 & 3)

- Levacher, K., Lawless S., Wade V. (2012) "An Evaluation and Enhancement of Densitometric Fragmentation for Content Slicing Reuse" *In the proceedings of the 21rst international conference on Information and Knowledge Management (CIKM'12)* (contribution to objective 1)

- Levacher, K., Lawless S., Wade V. (2011) "A Proposal for the Evaluation of Adaptive Content Retrieval, Modification and Delivery" *In the proceedings of the first Workshop on Personalised Multilingual Hypertext Retrieval (PMHR'11), in conjunction with the conference Hypertext and Hypermedia (HT'11)* (contribution to objective 2)

- Levacher, K., Hynes E., Lawless S., O'Connor A., Wade V. (2009) "A Framework for Content Preparation to Support Open-Corpus Adaptive Hypermedia" *In the proceedings of the International Workshop on Dynamic and Adaptive Hypertext (DAH'09), in conjunction with the conference Hypertext and Hypermedia (HT'09)* (contribution to objective 2)

6.3 Future Work

6.3.1 Introduction

The research described in this thesis has presented a novel approach to open-corpus content reuse within AHSs, through their provision via an independent slicer service capable of harvesting, customising and delivering such resources in content packages meeting arbitrary niche content requirements. Results obtained from evaluations presented in chapter 5 suggest this approach could be successfully used as a AHS content provision service within various use case scenarios. These promising results however, also offer several research opportunities for future work, extending potential capabilities of the service, as well as addressing limitations discovered during this research. The remaining sections of this chapter provides an overview of the main areas in which future research could be undertaken.

6.3.2 Slicing Content/AHS Agnosticity

As explained in section 2.3.4, the diversity of open corpus resources available on the WWW is very wide. A content/AHS agnostic slicer should hence ideally provide slicing services for the widest range of resources targetable regardless of content format, language etc. In order to focus the evaluation of this research upon the overall approach to open corpus reuse via slicing techniques, purposely narrowed down the range of resources targetable by the slicer prototype, implemented for this research, to any HTML based open corpus resources available within the English language section 3.3.2.

Hence, following the validation of this approach to reuse, the most obvious initial extension of this research would consist in extending Slicepedia to provide ***multilingual slicing*** services. To achieve this goal, components of the existing prototype could be replaced, as an initial step, by multilingual alternatives as proposed within chapter 4. If the use of DCF fragmentation algorithms is to be extended for the purpose of multilingual slicing, the impact of parallel multilingual fragmentation precision decreases (with respect to fragment sizes as well as specific

language combinations discovered in section 5.2.5) over the overall slicing performance of a multilingual slicer prototype should be investigated.

Furthermore, as Slicepedia only targeted HTML resources, future research should investigate extending the open corpus reuse approach proposed by this research to support ***multi-format slicing***. This would include extending slicing principles to other popular textual formats such as PDF, Text and Word files, as well as more multimedia resources such as images, audio and video file formats [Ketterl2008, Yang2012].

Nevertheless, and more importantly, the ***content type dependency limitations*** of DCF algorithms, discovered during the analysis carried out in section 5.2, should be investigated in priority by future research if the use of this algorithm is to be consolidated for the purposes of content/AHS agnostic slicing in general. The poor performance of these algorithms measured over resources with low levels of editorial control represents an important limitation for mainstream content/AHS agnostic slicer deployments. The sudden increase in volume of user generated content produced (such as forum, posts, tweets etc.) over recent years [Agichtein2008] further accentuates the need to resolve this issue in priority. As mentioned in section 4.3.3.4.2, during the process of carrying out this research, the original DCF algorithm implementation proposed by [Kohlschutter2008a] has now being released as open source software, as well as other implementations too [Pomikalek2011]. For this reason, the content type analysis performed in section 5.2 should be initially repeated using this implementation instead, in order to confirm results presented within this research. If this dependency is confirmed, slicing limitations caused by this issue could be addressed by either i) improving DCF fragmentation itself in order to perform equally regardless of content type, or ii) involving open corpus page classification methods (as described in section 5.2.4.2) prior to fragmentation.

6.3.3 Semantic Slicing

As presented in section 2.4.4, many approaches have been developed in recent years, which aim to analyse or segment resources based upon their semantics. However, at

the time this research was been carried out, literature covering content fragmentation techniques (which was identified as a critical component of a slicer) did not appear to provide a clear account related to the benefits and disadvantages of different approaches, nor did it offer evaluations upon algorithm characteristics of importance for the purpose of slicing. As a result, research upon the reuse of open-corpus resources through slicing, heavily focused upon the benefits obtained through the right-fitting performed by slicers from the point of view of structural fragmentation (section 2.4.3) as opposed to semantic analysis. For this reason, a significant area of future research, extending the investigation carried out within this thesis, could build upon the slicer architecture presented in this document and instead explore in more depth semantic analysis aspects of this service.

An obvious extension of this work, could involve investigating ***AHS driven real-time semantic analysis*** of resources targeted for reuse. As for the content fragmentation aspect of slicing, since one of the design requirements of this thesis involved supporting possible third party institution collaborations, the process of structurally fragmenting and semantically analysing open-corpus resources ended up being performed a-priori of niche content requirement provision from AHSs. However, in the event this requirement is necessary, and as mentioned in section 4.2, both steps could be performed after the receipt of niche content requirements from AHSs. Hence the semantic analysis of resources could be performed specifically based upon the semantic needs of individual AHSs as opposed to generic potential needs.

A second semantic related area to be explored, and addressing an important limitation of the system developed for this research, would involve investigating ***content/AHS agnostic resource type detection*** analysis techniques for open-corpus documents. As mentioned in [Brusilovsky2007h], AHSs make extensive usage of annotation-specific information describing the *type* of documents presented to users (i.e: problem statement, example, theory etc.). The CAS offered by Slicepedia however currently does not offer this right-fitting variable to slice consumers. The lack of such information about slices currently requires slice consumers to infer this information based upon the type of open corpus resources targeted by such

systems. In the experiment carried out in section 5.5 for example, the AMAS AHS designers targeted mostly "SQL How To" online resources (such as tutorials) for slicing, which contain mostly SQL examples. AMAS designers hence, inferred the majority of slices received would be of this *type* and therefore recomposed these slices accordingly within their platform. Clearly this information should be produced by the slicer itself in order to provide a more fine-grained description of slices available for recomposition. Hence, the ability to detect the *type* characteristics of open corpus sections, fully-automatically and without knowing in advance what resources are targeted for slicing, would enable slice consumers to select and sequence slices according to these characteristics.

Finally, as results obtained from this research suggest the process of slicing open-corpus content affects the reading quality of native resources targeted for reuse, future investigations in semantic slicing should clearly attempt to ***reduce reading quality decreases***. This issue could be addressed by taking into account anaphora resolution as proposed by [Ganguly2011], as well as narrative cohesion [Hargood2011] of content generated as part of the slicing process, which could improve the reading flow of content produced. Content combination decisions could also take into account the semantic combinability of individual fragments merged together into slices, in order to reduce any decrease in understandability.

6.3.4 Alternative Slice Generation Techniques

Aspects concerning the slice generation unit of the slicing architecture presented in this research could also be explored. As mentioned in section 4.3.4.4, the generation of slices performed by the Slicepedia prototype, combines individual fragments originating only from identical native resources sources and only in increasing sequential order. Future research could hence explore aspects concerning ***non-sequenced multi-source slice*** generation procedures. A slicer implemented using such techniques would posses a higher recompositional capacity than Slicepedia and produce slices through the combination of individual fragments originating from various sources, in orders not necessarily following the initial sequence in native

resources.

The area of ***federated slicing*** introduced in section 4.4 could also be explored. Different levels of slicing federation could be investigated, in which Slicepedia would hand over various steps of the slicing process to other alternative independent institutions by taking full advantage of the web of slices. This work would permit various institutions to collaborate together and remove the need for a central slicing authority, which would increase the volume of slices available to the AHS community.

6.3.5 Slicer-Based Next Generation AHSs

Finally, the ability offered by slicers for content to be supplied on demand to AHSs, based upon niche content requirements unknown in advance by slicing services, opens the possibility for new opportunities with respect to the development of AHS platforms. As referred by [Steichen2011a], Slicer based content supply services, remove the need for AHSs to predict in advance the quantities as well as specificities of content to be produced for individual users. Hence, novel ***open and user-driven AHSs*** could be implemented, similar in style to the use-case scenario presented in section 3.3.2.1, in which users request personalised experiences to be produced within a more open range of possibilities. In such AHSs, the content used within presentations would not be determined in advance of users using the AHS but instead would be identified and selected in real time, (and at the last minute) based upon individual user needs and preferences.

Glossary

V_{max} text density threshold value.

W_x word wrapping.

ρ_{τ_x} ratio of circumference of circle to its diameter.

CAS (introduced in section 3.3.3.1)

The resulting overall array of possible adjustments (such as the extent of control over granularity, style and annotation) a slicer can offer upon an open-corpus resource is referred to as a slicers CAS .

SCI Content

Self-Contained and Independent content refers to content packages which are both concise and self-sufficient. They are concise in the sense that they do not contain any superfluous content, unnecessary or detrimental for the proper reuse of such resources. What is considered superfluous content can vary based upon the context of reuse of each resource. This could include navigation bars, menus or advertisement contained within the original native resource, as well as content not related to a particular chosen topic. SCI content is self-sufficient in the sense that it can be reused or understood without the need of additional external content. .

A-priori Content Production Model (introduced in section 2.3.3)

This thesis defines an a-priori content supply approach as a supply model in which the content requirements needs of AHSs are known to the content

supply model a-priori of any request, and resources are produced to match these requirements at design time .

Adjusted Random Index (introduced in section 5.2.2.1)

The Rand Index [Strehl2003] measures the similarities between two clustering methods by determining the number of agreements within two vectors. Within the context of this analysis, these vectors represent the fragment each particular tag belongs to. The ARI [Yeung2001] is simply a normalised extension of the Rand measure with agreements between clusters rated using values ranging from zero (no agreement) to one (perfect agreement) .

Boilerplate (introduced in section 2.4.4.4)

Different definitions of what constitutes boilerplate content have been proposed based upon the application area they are been used in. It is usually vaguely referred to as non-informative parts outside of the main content of a page, repeated between web pages and typically machine generated [Pomikalek2011]. The CleanEval competition [Baroni2008], which is the most popular benchmark used for this task, identifies for example boilerplate content specifically as consisting of navigation, list of links, copyright notices, advertisement, forms, duplicate material etc... .

Closed Corpus Resources (introduced in section 2.3.3)

The definition of what constitutes closed corpus resources within this thesis refers to that given by [Brusilovsky2007h], and is formalised as follows: Closed corpus content consist of *restricted sets* of *proprietary* resources, produced by *identifiable sets of authors* within a *controlled environment.* The *restricted* nature of these resources refers to the ease in identifying the total quantity of resources contained within a corpus, since the volume of resources available is generally fix. Closed corpus resources are *predictable* and *static* in the sense that documents and relationships between these documents possess pre-determined content as well as a common structure, known at design time by systems consuming these resources, which does not evolve much over time. In most

cases, closed corpus documents and systems consuming these resources are both produced by the same group of individuals .

Content Focus (introduced in section 3.3.3)

Relationship between an arbitrary focal point within a piece of content, and the sizing of the area to which it belongs. A content package correctly focused on a topic, for example, would only contain content about the topic in question. Content considered unrelated to that topic would be removed. In other words, the sizing of the content should match the area covered by this focal point .

Content Relevancy (introduced in section 3.3.2)

In the context of this research, relevancy depends upon the identification method selected by the designer of the slicer. What constitutes relevant content hence could go from a resource simply containing a list of arbitrary keywords in the case of traditional IR resource identification, up to a resource meeting strict topical threshold criteria in the case of more sophisticated focus crawling approaches .

Content Right-Fitting

Throughout this thesis, the process of right-fitting refers to the production of content, resulting in the customisation of preliminary native resources, meeting an arbitrary set of niche content requirements. The activity of right-fitting is a subset of content slicing, which additionally includes content identification, harvesting and delivery. Right-fitting more specifically involves the fragmentation, annotation, identification of pertinent parts and modification of native resources. Words such as customising or tailoring are used interchangeably within this document to refer to this concept. The right-fitting of native resource should lead to SCI content .

Content Slicing (introduced in section 3.3.2)

Following i) the receipt of a set of niche content requirement specifications from an individual AHS, the process of slicing content involves the ability to ii) identify large volumes of resources, available on the WWW, of potential

reuse relevancy based upon these requirements. Resources which are deemed relevant should be subsequently iii) harvested, iv) right-fitted and v) delivered to individual AHSs, as content packages immediately ready for reuse .

Content Type (introduced in section 3.3.3)

Content type in this research refers to content of types such as encyclopaedia articles, news articles, product and forum pages etc...) .

Content/AHS Agnostic Technique (introduced in section 3.3.2)

A slicer is content/AHS agnostic in the sense that content is produced i) for an non-established set of AHSs, built independently of the slicer, using techniques which do not require any predefined knowledge of either ii) the resources targeted for reuse or iii) the reuse purposes (i.e expressed as niche content requirements). The aim of such technique is to embrace the full diversity and volume of open corpus resources available to the AHS community by managing the unpredictability and variability of these resources and AHSs content requirements .

Content/AHS Specific Technique (introduced in section 2.3.4.4)

Within this thesis, content/AHS specific techniques refer to reuse approaches in which i) specific open corpus repositories, known at design time, are targeted for reuse in order to serve ii) predetermined niche content requirements for iii) an established set of AHSs .

Federated Slicing (introduced in section 4.4)

Federated slicing refers to a slicing process whereby the pre-fragmentation and/or post-fragmentation modules within the slicing pipeline are distributed across different independent institutions and the slicing data output by these modules published and shared upon the web of slices .

Fragmentation Scope (introduced in section 2.20)

Within the context of this research, fragmentation scope refers to the total set of pages targetable by a method over any time. The fragmentation scope of an approach is decomposed into three sub properties consisting in the: a)

supported content format(s), b) targetable range of pages and c) adjustability. While the content format property refers to the type of pages which an approach can target or fragmentation, the targetable range refers to the collection of pages belonging to the supported format, which can be fragmented for a given algorithm configuration. Adjustability refers to the degree of effort involved in moving from one target range of pages to another .

Granularity (introduced in section 3.3.3)

Within this research, granularity refers to the combination of both the sizing of content and its focus .

Independent AHS

From the point of view of the slicer, Independent AHSs consist of a 1/ non established set of AHSs, 2/ built independently of the slicer with 3/ no predefined knowledge of each reuse purposes/intentions (including both 1) adaptation techniques AND 2) use cases) .

Native Resources (introduced in section 3.2.2)

Native resources within the context of this research refers to resources harvested from the web without any modification .

Niche and Broad Content Requirements (introduced in section 2.3.3)

Throughout this thesis, niche content requirements of AHSs, also mentioned as right-fitting requirements, refer to the combination of specific resource attributes, which one individual AHS requires in order to support an arbitrary niche personalisation experience of a individual user. Such niche content requirements can consist of content packages i) consisting of five paragraphs in length, ii) covering topic τ, iii) written in language λ, iv) with beginner reading difficulty, and v) possessing annotations pointing to all personalities mentioned within the text. Such requirements are by nature very unpredictable since they can potentially cover a wide range of possible requirement combinations Niche content requirements are defined in opposition to *broad content requirements*,

which consist of general resource requirements shared by groups of AHSs such as resources within a broad domain(i.e history) of media type (i.e video resources) etc....

On-demand Content Production Model (introduced in section 3.3.2)

The on-demand nature of slicing, within the context of this research, refers in particular to the dynamics involved between the slice consumer and slicer rather than the entire content slicing process itself .

Open Corpus Resources (introduced in section 2.3.4)

Within this research, open corpus content consists of *unrestricted sets* of web resources, produced by *authors not necessarily identifiable* within an *uncontrolled environment.* The *unrestricted* nature of these resources refers to the difficulty in identifying the total quantity of resources contained within a corpus, since the volume of resources available is generally expanding. Open corpus resources are *un-predictable* and *variable* in the sense that documents and relationships between these documents do not necessarily possess pre-determined content or common structure, known at design time by systems consuming these resources, and can evolve over time. In most cases, open corpus documents and systems consuming these resources are usually produced by different group of individuals .

Recomposition (introduced in section 3.3.2)

The support for the recomposition of slices within various independent AHSs, as part of this research, refers specifically to the ability for open corpus content to be correctly 1) selected and 2) right-fitting, as well as the ability to access resulting slices as part of 3) multiple search and delivery channels .

Reuse Improvement (introduced in section 1.2)

By open corpus reuse improvement, this thesis focuses upon the ability of AHSs to request open-corpus resources i) without any pre-determined conditions (regarding the source or type of open corpus requested) ii) as right-fitted

content packages, iii) based upon a variety of possible AHS content requirement specifications, undetermined in advance by the slicer .

Scalable Slicing (introduced in section 3.3.2)

A scalable content model, within the context of this research, is scalable with respect to three overall dimensions, namely it should guarantee the provision of i) large volumes of content, ii) at low costs, iii) suitable for multiple AHSs performing arbitrary activities .

Semantic Analysis (introduced in section 4.3.3.5)

Semantic analysis in this these refers to both traditional synthetic analysis (POS, morphological analysis etc) and/or Information Extraction (IE) algorithms (named entity recognition, relationship extraction) .

Sizing (introduced in section 3.3.3)

Content sizing simply refers to the size of a particular content package. This size might be expressed by the number of words or paragraphs contained within the text .

Slice (introduced in section 3.3.2.1)

A slice consists of customised content produced from a set of fragment(s) (originating from pre-existing document(s)) assembled, combined with appropriate meta-data and right-fitted to the specific content requirements of a slice consumer, with various application-specific content-reuse intentions, different (or not) from the original author intentions. Slices can contain other slices and can be reused or re-composed with many slices. .

Slice Consumer (introduced in section 3.3.2.1)

A slice consumer consists of any client reusing slices produced by a slice consumer. .

SliceHit (introduced in section 4.3.4.1)

A SliceHit object consists of a blueprint for slices. It contains a description of all the attributes and information necessary (fragments and meta-data) for

Slicepedia to construct a slice. It can be stored by slice consumers and later on be forwarded back to Slicepedia to convert it into slices .

Slicing Data (introduced in section 4.2.3)

Slicing data refers to both the fragments and meta-data stored within Slicepedia and used to generate slices are referred to as slicing data .

Structural Fragmentation and Regions of a Page

This thesis considers coherent regions as individual parts of a single web document, showing information about one or more related items when it is rendered in a web browser [Sleiman2012]. Such items can be data records, information about products, pieces of news, headers with navigation menus, footers with contact and corporate information, or sidebars with advertisements, to mention a few examples. Structural fragmentation hence refers to the ability to identify such regions within a page and extracting each one as individual content packages called fragments..

Web of Slices

A *web of slices* would consist of a network representing relationships between meta-data and fragments, leading to a range of potential slice content packages. The web of data, or so-called linked data, represents relationships between machine readable concepts, while the web of documents (i.e: WWW) represents relationships between published documents directly consumable by humans. The web of slices would hence be intermediary to the web of data and documents in the sense that it would contain more than data (i.e fragments), but less than published documents since fragments and meta-data represent potential slice outputs. .

Bibliography

[Aduriz2013] Itziar Aduriz and Arantza Diaz De Ilarraza. Morphosyntactic disambiguation and shallow parsing. *Anuario del Seminario de Filología Vasca*, pages 1–21, 2013.

[Aggarwal2001] Charu C. Aggarwal, Fatima Al-Garawi, and Philip S. Yu. Intelligent crawling on the World Wide Web with arbitrary predicates. In *WWW'01: Proceedings of the 10th World Wide Web Conference*, pages 96–105, New York, New York, USA, 2001. ACM Press.

[Agichtein2008] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In *Proceedings of the international conference on Web search and web data mining*, page 183, New York, New York, USA, 2008. ACM Press.

[Ahmadi2008] Hamed Ahmadi and Jun Kong. Efficient web browsing on small screens. In *AVI '08: Proceedings of the working conference on Advanced Visual Interfaces*, page 23, New York, New York, USA, 2008. ACM Press.

[Ahn2007] Jae-wook Ahn, P Brusilovsky, Jonathan Grady, Daqing He, and Sue Yeon Syn. Open User Profiles for Adaptive News Systems : Help or Harm ? In *WWW'07: Proc. of the 16th int. conf. on World Wide Web*, pages 11–20, 2007.

[Alassi2012] Derar Alassi and Reda Alhajj. Effectiveness of template detection on

noise reduction and websites summarization. *Information Sciences*, 219:41–72, January 2012.

[Alcic2011] Sadet Alcic and Stefan Conrad. Page segmentation by web content clustering. In *WIMS '11: Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, number 3, page 1, New York, New York, USA, 2011. ACM Press.

[Alfonseca2002] Enrique Alfonseca and Suresh Manandhar. An Unsupervised Method for General Named Entity Recognition and Automated Concept Discovery. In *1st International Conference on General WordNet*, 2002.

[Anam2011] Rajibul Anam, Chin Kuan Ho, and Tek Yong Lim. Web Content Adaptation for Mobile Devices : A Greedy Approach. *IJNCAA'11: International Journal of New Computer Architectures and their Applications*, 1(4):1027–1042, 2011.

[Anderson2006] Chris Anderson. *The Long Tail: Why the Future of Business is Selling Less of More.* Hyperion Books, 2006.

[Antonacopoulos2007a] A Antonacopoulos, B Gatos, and D Bridson. Page Segmentation Competition. *ICDAR'07: 9th Int. Conf. on Document Analysis and Recognition*, 2007.

[Arias2009b] Javier Arias, Koen Deschacht, and Marie-Francine Moens. Language Independent Content Extraction from Web Pages. In *DIR'09: Proceedings of the 9th Dutch-Belgian Information Retrieval workshop*, pages 50 – 55, 2009.

[Babashzadeh2013] Atanaz Babashzadeh, Jimmy Huang, and Mariam Daoud. Exploiting semantics for improving clinical information retrieval. In *SIGIR '13: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, page 801, New York, New York, USA, 2013. ACM Press.

[Bailey2006] Chris Bailey, T Mohd, C Hugh, Mohd T Zalfan, and Hugh C Davis. Panning for gold : designing pedagogically-inspired learn- ing nuggets. *Educational Technology and Society*, 2006.

[Ballantine2004] James Ballantine. *Topic Segmentation in Spoken Dialogue.* PhD thesis, Macquarie University, 2004.

[Bar-Yossef2002a] Ziv Bar-Yossef and Sridhar Rajagopalan. Template Detection via Data Mining and its Applications. *WWW'02: Proceedings of the 11th international conference on World Wide Web*, 2002.

[Baroni2008] Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. CleanEval : A Competition for Cleaning Webpages. In *LREC08: Proceedings of the Sixth International Language Resources and Evaluation*, pages 638–643, 2008.

[Beitzel2004] Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, David Grossman, and Ophir Frieder. Hourly analysis of a very large topically categorized web query log. In *SIGIR '04 Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, page 321, New York, New York, USA, 2004. ACM Press.

[Bergmark2002] Donna Bergmark, Carl Lagoze, and Alex Sbityakov. Focused Crawls, Tunneling , and Digital Libraries. In *ECDL 02: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, 2002.

[BernersLee1994a] T. Berners-Lee, Robert Cailliau, N. Pellow, and A. Secret. The World-Wide Web Initiative. Technical report, 1994.

[BernersLee2006c] Tim Berners-Lee, Nigel Shadbolt, and Wendy Hall. The Semantic Web Revisited. *IEEE Intelligent Systems*, 2006.

[Bernerslee2006] Tim Berners-lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings*

of the 3rd International Semantic Web User Interaction Workshop, number i, 2006.

[Bikel1997] Daniel M Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble : a High-Performance Learning Name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, 1997.

[Billsus2000] Daniel Billsus, Michael J. Pazzani, and James Chen. A learning agent for wireless news access. In *Proceedings of the 5th international conference on Intelligent user interfaces.*, pages 33–36, New York, New York, USA, 2000. ACM Press.

[Blei2001] David M Blei and Pedro J Moreno. Topic Segmentation with an Aspect Hidden Markov Model. In *SIGIR'01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001.

[Bontcheva2004] Kalina Bontcheva and Vania Dimitrova. Examining the Use of Conceptual Graphs in Adaptive Web-Based Systems That Aid Terminology Learning. *International Journal on Artificial Intelligence Tools*, 13:299–332, June 2004.

[Boulis2005] Constantinos Boulis. *Topic Learning in Text and Conversational Speech.* PhD thesis, University of Washington, 2005.

[Branco2009] Antonio Branco, Francisco Costa, Eduardo Ferreira, Pedro Martins, Filipe Nunes, and Sara Silveira. LX-Center : a center of online linguistic services. In *ACLDemos'09: Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, number August, pages 5–8, 2009.

[Brill1992] Eric Brill. A simple rule-based part of speech tagger. In *ANLC '92 Proceedings of the third conference on Applied natural language processing*, 1992.

[Brin1998] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale

Hypertextual Web Search Engine. In *WWW'98: - Seventh International World-Wide Web Conference*, 1998.

[Brusilovsky1996] Peter Brusilovsky, Elmar Schwarz, and Gerhard Weber. ELM-ART : An Intelligent Tutoring System on World Wide Web. In *ITS'96: Proc. of Third Interna- tional Conference on Intelligent Tutoring Systems*, 1996.

[Brusilovsky1996a] Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, July 1996.

[Brusilovsky1998a] Peter Brusilovsky and Leonid Pesin. Adaptive Navigation Support in Educational Hypermedia: An Evaluation of the ISIS-Tutor. *Journal of Computing and Information Technology*, 1998.

[Brusilovsky1998c] Peter Brusilovsky, John Eklund, and Elmar Schwarz. Web-based Education for All : A Tool for Development Adaptive Courseware Web-based Education. In *WWW'98: Proceedings of Seventh International World Wide Web Conference*, volume 30, pages 291–300, 1998.

[Brusilovsky2001] Peter Brusilovsky. Adaptive Hypermedia. In *User modeling and user-adapted interaction*, pages 87–110, 2001.

[Brusilovsky2004] Peter Brusilovsky, G Chavan, and R Farzan. Social adaptive navigation support for open corpus electronic textbooks. In *AH'04: Proc. of the 3rd int. conf. on Adaptive Hypermedia and Adaptive Web Based Systems*, volume 3137, 2004.

[Brusilovsky2007b] Peter Brusilovsky. Adaptive Navigation Support. In *The Adaptive Web: Methods and Strategies of Web Personalisation. Lecture Notes in Computer Science*, chapter 8, pages 263 – 290. 2007.

[Brusilovsky2007h] Peter Brusilovsky and Nicola Henze. Open Corpus Adaptive Educational Hypermedia. In *The Adaptive Web: Methods and Strategies*

of Web Personalisation. Lecture Notes in Computer Science, chapter 22. 2007.

[Bunt2007] Andrea Bunt, Giuseppe Carenini, and Cristina Conati. Adaptive Content Presentation for the Web. In *The Adaptive Web: Methods and Strategies of Web Personalisation. Lecture Notes in Computer Science*, chapter 13, pages 409–432. 2007.

[Burget2007] Radek Burget. Automatic Document Structure Detection for Data Integration. In *BIS'07: Proceedings of the 10th International Conference on Business Information Systems*, pages 391–397, 2007.

[Buttler2004a] David Buttler. A Short Survey of Document Structure Similarity Algorithms. In *ICOMP: Proceedings of the 5th international conference on internet computing*, 2004.

[Cai2003a] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Extracting Content Structure for Web Pages based on Visual Representation. In *APWeb'03: Proc. of the 5th Asia-Pacific web conf. on Web technologies and applications*, volume 2642 of *Lecture Notes in Computer Science*, page 596, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Cai2004] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Block-based web search. In *SIGIR '04: Proceedings of the 27th annual International Conference on Research and Development in Information Retrieval*, page 456, New York, New York, USA, 2004. ACM Press.

[Cano2013] Amparo E. Cano, Andrea Varga, Matthew Rowe, Fabio Ciravegna, and Yulan He. Harnessing linked knowledge sources for topic classification in social media. In *HT '13 - Proceedings of the 24th ACM Conference on Hypertext and Social Media*, number May, pages 41–50. ACM Press, 2013.

[Cao2008] Donglin Cao, Xiangwen Liao, Hongbo Xu, and Shuo Bai. Blog Post and Comment Extraction Using. In *AIRS'08 Proceedings of the 4th Asia*

information retrieval conference on Information retrieval technology, pages 298–309, 2008.

[Carmona2002] Cristina Carmona, David Bueno, and Ricardo Conejo. SIGUE: Making Web Courses Adaptive. In *HT'02: Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 376–379, 2002.

[Carroll2005] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. *WWW '05: Proceedings of the 14th international conference on World Wide Web*, page 613, 2005.

[Cerf1974] Vinton Cerf, Yogen Dalal, and Carl Sunshine. Specification of the Internet Transmission Control Program, 1974.

[Chakrabarti1999] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11-16):1623–1640, May 1999.

[Chakrabarti2002] Soumen Chakrabarti, Kunal Punera, and Mallela Subramanyam. Accelerated Focused Crawling through Online Relevance Feedback. In *WWW'02: Proceedings of the Eleventh International World Wide Web Conference*, page 148, New York, New York, USA, 2002. ACM Press.

[ChakrabartiD2007a] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. Page-level template detection via isotonic smoothing. *WWW'07: Proc. of the 16th int. conf. on World Wide Web*, pages:61–70, 2007.

[ChakrabartiD2008a] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. A Graph-Theoretic Approach to Webpage Segmentation. In *WWW'08: Proc. of the 17th International World Wide Web Conference*, pages 377–386, 2008.

[ChakrabartiD2010a] Deepayan Chakrabarti and Rupesh R Mehta. The Paths More Taken : Matching DOM Trees to Search Logs for Accurate Webpage

Clustering. In *WWW'10: Proc. of the 19th International World Wide Web Conference*, 2010.

[Chang2006] Chia-Hui Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan. A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, October 2006.

[ChenitiBelcadhi2006] Lilia Cheniti-Belcadhi, Nicola Henze, and Rafik Braham. Implementation of a Personalized Assessment Web Service. In *ICALT'06: Proc. of the Sixth International Conference on Advanced Learning Technologies, 2006*, pages 586–590. Ieee, 2006.

[Collobert2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 2011.

[Conlan2002] Owen Conlan, Vincent Wade, Catherine Bruen, and Mark Gargan. Multi-model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning. In *AH'02: Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 100–111, London, UK, 2002. Springer-Verlag.

[Conlan2004a] Owen Conlan and Vincent Wade. Evaluation of APeLS - An Adaptive eLearning Service based on the Multi-model , Metadata-driven Approach. In *AH'04: Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2004.

[Conlan2013F] O. Conlan, T. Staikopoulos, C. Hampson, S. Lawless, and I O'Keeffe. The Narrative Approach to Personalisation. *New Review of Hypermedia and Multimedia*, 2013.

[Crescenzi1998a] V Crescenzi and Mecca Giansalvatore. Grammars have exceptions. *Information Systems*, 1998.

[Crescenzi2001] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. RoadRunner : Towards Automatic Data Extraction from Large Web

Sites. In *VLDB '01 Proceedings of the 27th International Conference on Very Large Data Bases*, 2001.

[DeBra1994a] P De Bra and R.D.J. Post. Searching for Arbitrary Information in the World Wide Web the Fish-Search for Mosaic. In *WWW'94: Proceedings of the 2nd World Wide Web Conference*, 1994.

[DeBra1997] Paul De Bra and Licia Calvi. Creating Adaptive Hyperdocuments for and on the Web. In *AACE'97: Proceedings of the AACE WebNet Conference*, 1997.

[DeBra1998] Paul De Bra and Licia Calvi. AHA a Generic Adaptive Hypermedia System. In *Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia*, 1998.

[DeBra1999f] Paul De Bra, Houben Geert-Jan, and Wu Hongjing. AHAM : A Dexter-based Reference Model for Adaptive Hypermedia. In *HT'99: Proceedings of the tenth ACM Conference on Hypertext and hypermedia*, pages 147–156, 1999.

[DeBra2006a] P. De Bra, David Smits, and Natalia Stash. Creating and Delivering Adaptive Courses with AHA ! In *EC-TEL'06: European Conference on Technology Enhanced Learning*, 2006.

[DeBra2006b] P. De Bra, David Smits, and Natalia Stash. The Design of AHA! In *HT'06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 133–133, 2006.

[Debnath2005c] Sandip Debnath, Prasenjit Mitra, Nirmal Pal, and C Lee Giles. Automatic Identification of Informative Sections of Web Pages. *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1233–1246, 2005.

[Dempster1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[Dieberger2002] Andreas Dieberger and Mark Guzdial. CoWeb - Experiences with Collaborative Web spaces. In Danyel Lueg, Christoph and Fisher, editor, *From Usenet to CoWebs - Interacting with Social Information Spaces*, pages 155 – 166. Springer, 2002.

[Diligenti2000] M Diligenti, F M Coetzee, S Lawrence, C L Giles, and M Gori. Focused Crawling Using Context Graphs. In *VLDB 00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 527–534, 2000.

[Dimitrova1998] Vanya Dimitrova and Darina Dicheva. "Who is who": the roles in an intelligent system for foreign language terminology learning. *British Journal of Educational Technology*, 29(1):47–57, January 1998.

[Dimitrova2003] Vania Dimitrova. STyLE-OLM : Interactive Open Learner Modelling. *International Journal of Artificial Intelligence in Education*, 13:35–78, 2003.

[Dolog2004] Peter Dolog, Nicola Henze, Wolfgang Nejdl, and Michael Sintek. The Personal Reader : Personalizing and Enriching Learning Resources Using Semantic Web Technologies. *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 85–94, 2004.

[Esser2012] Daniel Esser, Daniel Schuster, Klemens Muthmann, Michael Berger, and Alexander Schill. Automatic Indexing of Scanned Documents - a Layout-based Approach. In *SPIE'12: IS&T/SPIE Electronic Imaging*, pages 1–8. International Society for Optics and Photonics, 2012.

[Fader2011] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying Relations for Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.

[Fielding1999] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol HTTP 1.1, 1999.

[Fielding2002] Roy T. Fielding and Richard N. Taylor. Principled Design of the Modern Web Architecture. *TOIT'02 ACM Transactions on Internet Technology*, 2(2):115–150, May 2002.

[Freitag2000a] Dayne Freitag and Andrew Mccallum. Information extraction with HMM structures learned by stochastic optimization. In *AAAI'00: Proc. of the 17th national conference on Artificial Intelligence*, 2000.

[Fumarola2011] Fabio Fumarola, Tim Weninger, Rick Barber, Donato Malerba, and Jiawei Han. Extracting General Lists from Web Documents : A Hybrid Approach. *Modern Approaches in Applied Intelligence*, pages 285–294, 2011.

[Ganguly2011] Debasis Ganguly, Johannes Leveling, and Gareth J F Jones. Utilizing sub-topical structure of documents for Information Retrieval Categories and Subject Descriptors. In *Utilizing sub-topical structure of documents for Information Retrieval*, pages 19–22, 2011.

[Gao2005] Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. Chinese Word Segmentation and Named Entity Recognition : A Pragmatic Approach. *Computational Linguistics*, 2005.

[Gao2013] Jianfeng Gao, Gu Xu, and Jinxi Xu. Query Expansion Using Path-Constrained Random Walks. In *SIGIR '13: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013.

[Gasparetti2004] Fabio Gasparetti and Alessandro Micarelli. Swarm Intelligence : Agents for Adaptive Web Search. In *ECAI'04: Proceedings of the 16th European Conference on Artificial Intelligence*, 2004.

[Ghorab2012] M. Rami Ghorab, Dong Zhou, Alexander OConnor, and Vincent Wade. Personalised Information Retrieval: survey and classification. *User Modeling and User-Adapted Interaction*, 23(4):381–443, May 2012.

[Gibson2005] D Gibson, K Punera, and Andrew Tmokins. The volume and evolution of web page templates. In *WWW'05: Proceedings of the international conference on the World Wide Web - Poster Session,*, 2005.

[Gibson2007a] John Gibson, Ben Wellner, and Susan Lubar. Adaptive Web-page Content Identification. In *WIDM'07: Web Information and Data Management*, pages 105–112, 2007.

[Gong2012] Yunfei Gong and Qiang Liu. Automatic web page segmentation and information extraction using conditional random fields. In *CSCWD'12: Proc. Int.Conf. on Computer Supported Cooperative Work in Design*, pages 334–340. Ieee, May 2012.

[GonzalezBarahona2006] Jesus M Gonzalez-barahona, Vania Dimitrova, Diego Chaparro, Chris Tebb, Teofilo Romera, Luis Canas, Julika Matravers, and Styliani Kleanthous. Towards Community-Driven Development of Educational Materials : The Edukalibre Approach. In *EC-TEL'06: First European Conference on Technology Enhanced Learning*, 2006.

[Gottron2008a] Thomas Gottron. Content Code Blurring: A New Approach to Content Extraction. In *DEXA'08: 19th International Conference on Database and Expert Systems Applications*, pages 29–33. Ieee, 2008.

[Grishman1997] R. Grishman. TIPSTER Architecture Design Document Version 2.3. Technical report, 1997.

[Gupta2005b] Suhit Gupta, Gail Kaiser, and Salvatore Stolf. Extracting context to improve accuracy for HTML content extraction. *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, 2005.

[Hammer1997] Joachim Hammer, Jason McHugh, and Hector Garcia-Molina. Semistructured Data: The TSIMMIS Experience. *First East European Symposium on advances in databases and information systems*, 1997.

[Hargood2011] Charlie Hargood, David E Millard, and Mark J Weal. Measuring Narrative Cohesion : A Five Variables Approach. In *AH'11: Narrative and Hypertext Workshop at Hypertext 11*, 2011.

[Hassan2005] Tamir Hassan and Robert Baumgartner. Intelligent Wrapping from PDF Documents. In V. Svatek and V. Snasel, editors, *RAWS'05*, pages 33–40, 2005.

[Hattori2007] Gen Hattori, Keiichiro Hoashi, Kazunori Matsumoto, and Fumiaki Sugaya. Robust Web Page Segmentation for Mobile Terminal Using Content-Distances and Page Layout Information. In *WWW 07: Proceedings of the 16th International Conference on WorldWideWeb*, pages 361–370, 2007.

[Haveliwala2003] T.H. Haveliwala. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, July 2003.

[Hearst1997] Marti A Hearst. TextTiling : Segmenting Text into Multi-paragraph Subtopic Passages. *Journal of Computational Linguistics*, 23(1):33–64, 1997.

[Henze2001a] Nicola Henze and Wolfgang Nejdl. Adaptation in Open Corpus Hypermedia. *IJAIED'01: Int. Journal of Artificial Intellligence in Education*, pages 325 – 350, 2001.

[Henze2002a] Nicola Henze and Wolfgang Nejdl. Knowledge Modeling for Open Adaptive Hypermedia. In *AH'02: Adaptive hypermedia and adaptive web-based systems*, pages 174–183, 2002.

[Henze2005] Nicola Henze. Personal Readers : Personalized Learning Object Readers for the Semantic Web 1. In *AIED'05: International Conference on Artificial Intelligence in Education*, 2005.

[Hepple2000] Mark Hepple. Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers. In *ACL'00:Proc.*

of the 38th Annual Meeting of the Association for Computational Linguistics, 2000.

[Hochbaum1985] Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operational Research*, 10(2):180–184, 1985.

[Hodgins2002] H Wayne Hodgins. The Future of Learning Objects. *e-Technologies in Engineering Education Learning Outcomes Providing Future Possibilities*, (August):76–82, 2002.

[Hogue2005] Andrew Hogue and David Karger. Thresher : Automating the Unwrapping of Semantic Content from the World Wide Web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 86–95, 2005.

[Holzinger2006] Wolfgang Holzinger, Bernhard Krupl, and Marcus Herzog. Using Ontologies for Extracting Product Features from Web Pages. *ISWC'06: The Semantic Web*, 2006.

[Hook1997b] Kristina Höök. Evaluating the Utility and Usability of an Adaptive Hypermedia System. In *IUI'97: Proc. of International Conference on Intelligent User Interfaces*, number Brusilovsky, pages 1–16, 1997.

[Hu2000] Jianying Hu, Ramanujan Kashi, and Gordon Wilfond. Comparison and Classification of Documents Based on Layout Similarity. *Journal on Information Retrieval*, 243:227–243, 2000.

[Hu2009] Yan Hu and Miao Miao. Research and Implementation on Multi-cues Based Page Segmentation Algorithm. In *CiSE'09: Conf. Computational Intelligence and Software Engineering, 2009*, pages 3–6, 2009.

[Hulpu2013] Ioana Hulpu, Conor Hayes, and Derek Greene. Unsupervised Graph-based Topic Labelling using DBpedia. In *WSDM'13: Proceedings of the sixth ACM International Conference on Web Search and Data Mining*, 2013.

[Hutchins2005] John Hutchins. The first public demonstration of machine translation : the Georgetown-IBM system , 7th January 1954. In *AMTA conference*, number January 1954, 1954.

[ICDAR2011] ICDAR Competition, 2011.

[ITU2013] ITU. The World in 2013: ICT Facts and Figures. Technical report, International Telecommunication Union, 2013.

[Irmak2006] Utku Irmak and Torsten Suel. Interactive wrapper generation with minimal user effort. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, page 553, New York, New York, USA, 2006. ACM Press.

[Jameel2013] Shoaib Jameel and Wai Lam. An unsupervised topic segmentation model incorporating word order. In *SIGIR '13: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 203, New York, New York, USA, 2013. ACM Press.

[Jamilah2010] Raja Jamilah, Raja Yusof, Roziati Zainuddin, and Mohd Sapiyan Baba. Qur'anic Words Stemming. *AJSE'10: Arabian Journal for Science and Engineering*, 35(2):37–49, 2010.

[Jansen2000] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36(2):207–227, March 2000.

[Kaasinen2000] Eija Kaasinen, Matti Aaltonen, Juha Kolari, Suvi Melakoski, and Timo Laakko. Two approaches to bringing Internet services to WAP devices. In *WWW'00: Proceedings of the 9th international World Wide Web conference on Computer networks*, volume 33, pages 231–246, June 2000.

[Kao2005b] Hung-yu Kao, Jan-ming Ho, and Ming-syan Chen. WISDOM: Web Intra-page Informative Structure Mining based on Document Object

Model. *TKDE - IEEE Transactions on Knowledge and Data Engineering*, 17, 2005.

[Karane2006a] Vieira Karane, Da Silva Altigran S., Pinto Nick, De Moura Edleno S., Cavalcanti João M. B., and Freire Juliana. A Fast and Robust Method for Web Page Template Detection and Removal. In *CIKM"06: Proceedings of the 15th ACM international conference on Information and knowledge managemen*, pages 258–267, 2006.

[Kavcic2004] A. Kavcic. Fuzzy User Modeling for Adaptation in Educational Hypermedia. In *Transactions on Systems, Man, and Cybernetics*, volume 34, pages 439–449, 2004.

[Kazantseva2011] Anna Kazantseva and Stan Szpakowicz. Linear Text Segmentation Using Affinity Propagation. In *EM-NLP'11: Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pages 284–293, 2011.

[Kelly2003] Diane Kelly and Jaime Teevan. Implicit Feedback for Inferring User Preference : A Bibliography Behavior Category. *ACM SIGIR Forum*, 37(2), 2003.

[Ketterl2008] Markus Ketterl, Johannes Emden, and Jorg Brunstein. Social Selected Learning Content Out of Web Lectures. In *HT'08: Proceedings of the 19th ACM conference on Hypertext and hypermedia*, pages 231–232, 2008.

[Kim2002b] Sanghee Kim, Harith Alani, Wendy Hall, Paul H Lewis, David E Millard, Nigel R Shadbolt, and Mark J Weal. Artequakt : Generating Tailored Biographies with Automatically Annotated Fragments from the Web. In *ECAI'02: Proc. of the 15th euro. conf. on Artificial Intelligence*, 2002.

[Kincaid1975] J.P. Kincaid, R.P. Fishburne, R.L. Rogers, and B.S Chissom. Derivation of New Readability Formulas (Automated Readability Index, Fog Count, and Flesch Reading Ease formula) for Navy Enlisted

Personnel. Technical report, Research Branch: Naval Air Station, Memphis, 1975.

[Kleinberg1999] Jon Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999.

[Knutov2009] Evgeny Knutov, P. De Bra, and Mykola Pechenizkly. AH 12 years later : a comprehensive survey of adaptive hypermedia methods and techniques. *New Review of Hypermedia and Multimedia*, (June 2012):37–41, 2009.

[Knutov2010] Evgeny Knutov, P. De Bra, and Mykola Pechenizkiy. Adaptation and Search : from Dexter and AHAM to GAF. In *HT'10: Proceedings of the 21st ACM conference on Hypertext and Hypermedia*, 2010.

[Kobilarov2009a] Georgi Kobilarov, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee. Media Meets Semantic Web How the BBC Uses DBpedia and Linked Data to Make Connections. *The semantic web: research and applications*, pages 723–737, 2009.

[Kobsa2001] Alfred Kobsa, Jürgen Koenemann, and Wolfgang Pohl. Personalised hypermedia presentation techniques for improving online customer relationships. *The Knowledge Engineering Review*, 16(02):111, December 2001.

[Kohlschutter2008a] Christian Kohlschütter and Wolfgang Nejdl. A Densitometric Approach to Web Page Segmentation. In *CIKM'08: Proc. of the 17th int. conf. on Information and knowledge management*, pages 1173–1182, 2008.

[Kohlschutter2010a] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate Detection using Shallow Text Features. In *WSDM'10: Proceedings of the 3rd int. conf. on Web Search and Data Mining*, 2010.

[Koidl2011] Kevin Koidl, Owen Conlan, and Vincent Wade. Towards User-centric Cross-site Personalisation. In *ICWE'11: In Proceedings of the 11th International Conference on Web Engineering*, pages 2–5, 2011.

[Kyriakopoulou2008] Antonia Kyriakopoulou. Text Classification Aided by Clustering : a Literature Review. *Tools in Artificial Intelligence*, 2008.

[Laender2002a] Alberto Laender, Berthier Ribeiro-Neto, Altigran Da Silva, and Juliana Teixeira. A Brief Survey of Web Data Extraction Tools. *SIGMOD'02: Special Interest Group On Management of Data*, 31, 2002.

[Lau2010] Jey Han Lau, David Newman, Sarvnaz Karimi, and Timothy Baldwin. Best Topic Word Selection for Topic Labelling. In *Proceedings of the 23rd International Conference on Computational Linguistics*, number August, pages 605–613, 2010.

[Lawless2008a] Séamus Lawless, Lucy Hederman, and Vincent Wade. OCCS : Enabling the Dynamic Discovery , Harvesting and Delivery of Educational Content from Open Corpus Sources. In *ICALT'08: Proc. of the Int. Conf. on Advanced Learning Technologies*, pages 676–678. IEEE Computer Society, 2008.

[Lawless2009] S Lawless. *Leveraging Content from Open Corpus Sources for Technology Enhanced Learning.* PhD thesis, Trinity College Dublin, 2009.

[Lee2005] Chang-Shing Lee, Zhi-Wei Jian, and Lin-Kai Huang. A fuzzy ontology and its application to news summarization. *IEEE Transactions*, 35(5):859–80, October 2005.

[Leoncini2012] Alessio Leoncini, Fabio Sangiacomo, Paolo Gastaldo, and Rodolfo Zunino. A Semantic-Based Framework for Summarization and Page Segmentation in Web Mining. In *Theory and Applications for Advanced Text Mining*, chapter 4. 2012.

[Levacher2011] Killian Levacher, Seamus Lawless, and Vincent Wade. A Proposal for the Evaluation of Adaptive Content Retrieval, Modification and Delivery. In *HT'11: Proc. of 22nd Conf. on Hypertext and Hypermedia*, 2011.

[Levacher2012d] Killian Levacher, Seamus Lawless, and Vincent Wade. Slicepedia: Automating the Production of Educational Resources from Open Corpus Content. In *EC-TEL'12: Proceedings of the 23rd ACM conference on Hypertext and social media*, 2012.

[Lieberman1995] Henry Lieberman. Letizia : An Agent That Assists Web Browsing. *IJCAI'95: Proceedings of the 18th international joint conference on Artificial intelligence*, 1995.

[Lin2002a] Shian-Hua Lin and Jan-Ming Ho. Discovering informative content blocks from Web documents. In *KDD'02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge Discovery and Data mining*, page 588, New York, New York, USA, 2002. ACM Press.

[Liu2000a] L Liu, C Pu, and W Han. XWRAP: an XML-enabled wrapper construction system for Web information sources. In *ICDE'00 - Proceedings of the 16th International Conference on Data Engineering*, 2000.

[Lloret2012] Elena Lloret and Manuel Palomar. Text summarisation in progress: a literature review. *Artificial Intelligence Review*, 37(1):1–41, April 2012.

[Lo2006] Lawrence Lo, Vincent Ng, Patrick Ng, and Stephen Chan. Automatic Template Detection for Structured Web Pages. In *CSCWD '06: Computer Supported Cooperative Work in Design*, pages 1–6. Ieee, May 2006.

[Lops2011] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor,

editors, *Recommender Systems Handbook*, pages 73–105. Springer US, Boston, MA, 2011.

[Louvan2009a] Samuel Louvan. Extracting the Main Content from HTML Documents. Technical report, Technische Universiteit Eindhoven, Eindhoven, 2009.

[Macqueen1996] J Macqueen. Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 233(233):281–297, 1996.

[Magdy2011] Walid Magdy and Gareth J.F. Jones. An efficient method for using machine translation technologies in cross-language patent search. In *CIKM '11: Proceedings of the 20th ACM international conference on Information and knowledge management*, page 1925, New York, New York, USA, 2011. ACM Press.

[Mahmud2007] Jalal Mahmud, Yevgen Borodin, and I.V Ramakrishnan. CSurf : A Context-Driven Non-Visual Web-Browser. In *WWW '07 Proceedings of the 16th international conference on World Wide Web*, volume 1, pages 31–40, 2007.

[Mangold2007] Christoph Mangold. A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, 2(1):23, 2007.

[Marek2007a] Michal Marek, Pavel Pecina, and Miroslav Spousta. Web Page Cleaning with Conditional Random Fields. In *Proceedings of the Fifth Web as Corpus Workshop, Incorporationg CleanEval*, volume 5, pages 1–8, 2007.

[Maxwell2013] K Tamsin Maxwell and W Bruce Croft. Compact Query Term Selection Using Topically Related Text. In *SIGIR '13 Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 583–592, 2013.

[Mccallum2003] Andrew Mccallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning*, 2003.

[Mehta2005] Rupesh R. Mehta, Pabitra Mitra, and Harish Karnick. Extracting semantic structure of web documents using content and visual information. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, page 928, New York, New York, USA, 2005. ACM Press.

[Mei2007] Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. Automatic labeling of multinomial topic models. In *KDD'07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 490, New York, New York, USA, 2007. ACM Press.

[Menczer2000] Filippo Menczer and Richard Belew. Adaptive Retrieval Agents : Internalizing Local Context and Scaling up to the Web. *Machine Learning*, pages 203–242, 2000.

[Meyer2011] Marek Meyer, Christoph Rensing, and Ralf Steinmetz. Multigranularity reuse of learning resources. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 7(1):1–23, January 2011.

[Micarelli2007a] Alessandro Micarelli, Fabio Gasparetti, Filippo Sciarrone, and Susan Gauch. Personalized Search on the World Wide Web. In *The Adaptive Web: Methods and Strategies of Web Personalisation. Lecture Notes in Computer Science*, pages 195–230. 2007.

[Micarelli2007b] Alessandro Micarelli and Fabio Gasparetti. Adaptive Focused Crawling. In *The Adaptive Web: Methods and Strategies of Web Personalisation. Lecture Notes in Computer Science*, pages 231–262. 2007.

[Millard2003] David Millard, Harith Alani, Sanghee Kim, Mark Weal, Paul Lewis, Wendy Hall, David De Roure, and Nigel Shadbolt. Generating Adaptive Hypertext Content from the Semantic Web. In *1st International Workshop on Hypermedia and the Semantic Web*, pages 1–9, 2003.

[Moura2010] Edleno S De Moura, David Fernandes, Berthier Ribeiro-neto, Altigran S Silva, and Marcos Andre Goncalves. Using Structural Information to Improve Search in Web Collections. *Journal of the American Society for Information Science and Technology*, 61(12):2503–2513, 2010.

[Nadeau2007] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, (1991):1–20, 2007.

[Nejdl2002] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Mikael Nilsson, Matthias Palm, and Tore Risch. EDUTELLA : A P2P Networking Infrastructure Based on RDF. In *WWW'02: Proceedings of the 11th international conference on World Wide Web*, 2002.

[OKeeffe2012] Ian O Keeffe, Athanasios Staikopoulos, Rachael Rafter, Eddie Walsh, Bilal Yousuf, Owen Conlan, and Vincent Wade. Personalized Activity Based eLearning. In *iKnow'12: Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, 2012.

[Pappas2012] Nikolaos Pappas, Georgios Katsimpras, and Efstathios Stamatatos. Extracting Informative Textual Parts from Web Pages Containing User-Generated Content Categories and Subject Descriptors. In *i-KNOW '12: Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, 2012.

[Pasternack2009a] Jeff Pasternack and Dan Roth. Extracting Article Text from the Web with Maximum Subsequence Segmentation. In *WWW'09:*

Proceedings of the 18th International Conference on World Wide Web, pages 971–980, 2009.

[Peng2004] Fuchun Peng. Chinese segmentation and new word detection using conditional random fields. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, pages 562–es, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[Plate2006] Carolin Plate, Nathalie Basselin, Alexander Kroner, Michael Schneider, Stephan Baldes, Vania Dimitrova, and Anthony Jameson. Recomindation : New Functions for Augmented Memories. In *AH'06: Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 1–10, 2006.

[Pomikalek2011] Jan Pomikalek. *Removing Boilerplate and Duplicate Content from Web Corpora*. PhD thesis, 2011.

[Porter1980] Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 1980.

[Prasad2007] R Prasad, P Natarajan, K Subramanian, S Saleem, and R Schwartz. Finding Structure in Noisy Text : Topic Classification and Unsupervised Clustering HMM based Topic Classification. *International Journal of Document Analysis and Recognition (IJDAR)*, pages 3–8, 2007.

[Ptaszynski2012] Michal Ptaszynski and Yoshio Momouchi. Part-of-speech tagger for Ainu language based on higher order Hidden Markov Model. *Expert Systems with Applications*, 39(14):11576–11582, October 2012.

[Raghunathan2010] Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. A Multi-Pass Sieve for Coreference Resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, number October, pages 492–501, 2010.

[Ramakrishnan2008] Cartic Ramakrishnan, Pablo N Mendes, Shaojun Wang, and Amit Sheth. Unsupervised Discovery of Compound Entities for Relationship Extraction. *Knowledge Engineering: Practice and Patterns*, pages 146–155, 2008.

[Ramaswamy2004] L Ramaswamy, A Lyengar, L Liu, and F Douglis. Automatic Detection of Fragments in Dynamically Generated Web Pages. In *WWW'04: Proc. of the 13th World Wide Web Conference*, volume pp, pages 443–454, 2004.

[Rao2012] G V Subba Rao and K Ramesh. Automatic Wrapper Generation for Search Engines Based on Visual Representation. *IJRCCT'12: International Journal of Research in Computer and Communication Technology*, 1(4):164–169, 2012.

[Robertson1994] S E Robertson and S Walker. Some for Simple Effective Approximations to the 2 Poisson Model Probabilistic Weighted Retrieval The. In *SIGIR '94 Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, number 1, 1994.

[Sah2009a] Melike Sah. *Semantic Linking and Personalization in Context.* PhD thesis, University of Southampton, 2009.

[Sah2010] Melike Sah and Vincent Wade. Automatic metadata extraction from multilingual enterprise content. In *CIKM '10: Proceedings of the 19th ACM international conference on Information and knowledge management*, page 1665, New York, New York, USA, 2010. ACM Press.

[Sah2012] Melike Sah and Vincent Wade. Automatic metadata mining from multilingual enterprise content. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 11:41–62, March 2012.

[Sah2013] M. Sah and W. Hall. A personalized semantic portal for enhanced

user support. *New Review of Hypermedia and Multimedia*, 19(1):25–60, March 2013.

[Schedl2008] Markus Schedl, Peter Knees, Tim Pohle, and Gerhard Widmer. Towards an Automatically Generated Music Information System via Web Content Mining. *Advances in Information Retrieval*, 2008.

[Schmid1995] Helmut Schmid. Improvements In Part-of-Speech Tagging With an Application To German. In *Proceedings of the ACL SIGDAT-Workshop*, pages 1–9, 1995.

[Sha2003] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, number June, pages 134–141, 2003.

[Shen2007] Libin Shen, Giorgio Satta, and Aravind Joshi. Guided Learning for Bidirectional Sequence Classification. In *ACL'07: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, number June, pages 760–767, 2007.

[Silva1998] Denise Pilar Da Silva, Rafaël Van Durm, Erik Duval, and Henk Olivié. Concepts and documents for adaptive educational hypermedia: a model and a prototype. In *HT'98: Proc. of the International Hypertext Conference Hypertext'98*, 1998.

[Skantze2010] Gabriel Skantze and Anna Hjalmarsson. Towards Incremental Speech Generation in Dialogue Systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 1–8, 2010.

[Sleiman2012] Hassan Sleiman and Rafael Corchuelo. A Survey on Region Extractors From Web Documents. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2012.

[Smith2003] A Serengul Guven Smith and A N N Blandford. MLTutor : An Application of Machine Learning Algorithms for an Adaptive Web-based Information System. *International Journal of Artificial Intelligence in Education*, 13:1–22, 2003.

[Smyth2003] Barry Smyth and Paul Cotter. Intelligent Navigation for Mobile Internet Portals. In *IJCAI'03: The 18th International Joint Conference on Artificial Intelligence*, 2003.

[Song2004] Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. Learning block importance models for web pages. In *WWW '04 Proceedings of the 13th international conference on World Wide Web*, page 203, New York, New York, USA, 2004. ACM Press.

[Song2011b] Fei Song, William M Darling, Adnan Duric, and Fred W Kroon. An Iterative Approach to Text Segmentation. *Advances in Information Retrieval*, pages 629–640, 2011.

[Sosnovsky2012] Sergey Sosnovsky, I Hsiao, and Peter Brusilovsky. Adaptation in the Wild: Ontology-Based Personalization of Open-Corpus Learning Material. In *EC-TEL'12: Proceedings of the 7th European Conference on Technology Enhanced Learning*, pages 425–431, 2012.

[Spengler2010] Alex Spengler and Patrick Gallinari. Document Structure Meets Page Layout : Loopy Random Fields for Web News Content Extraction. In *DocEng '10 Proceedings of the 10th ACM symposium on Document engineering*, number Section 6, pages 151–160, 2010.

[Speretta2005a] M Speretta and S Gauch. Personalized search based on user search histories. In *Proceedings of the International Conference on Web Intelligence*, 2005.

[Staikopoulos2012] Athanasios Staikopoulos, Ian O Keeffe, Rachael Rafter, Eddie Walsh, Bilal Yousuf, Owen Conlan, and Vincent Wade. AMASE : A framework for composing adaptive and Personalised Learning Activities

on the Web. In *ICWL'12: Proceedings of the 11th Int. Conf. on Web-based Learning, Sinaia, Romania, September 2-4, 2012*, 2012.

[Staikopoulos2013] Athanasios Staikopoulos, Ian O Keeffe, Rachael Rafter, Eddie Walsh, Bilal Yousuf, Owen Conlan, and Vincent Wade. AMASE: A framework for supporting personalised activity-based learning on the web. *ComSIS'13: International Journal of Computer Science and Information Systems*, 2013.

[Stamou2008] Sofia Stamou and Alexandros Ntoulas. Search personalization through query and page topical analysis. *User Modeling and User-Adapted Interaction*, 19(1-2):5–33, September 2009.

[Steichen2009a] Ben Steichen, Seamus Lawless, Alexander O Connor, and Vincent Wade. Dynamic Hypertext Generation for Reusing Open Corpus Content. In *HT'09: Proceedings of the 20th ACM conference on Hypertext and hypermedia, Torino, Italy*, 2009.

[Steichen2010b] Ben Steichen and Vincent Wade. Adaptive Retrieval and Composition of Socio-Semantic Content for Personalised Customer Care. *Proceedings of International Workshop on Adaptation in Social and Semantic Web*, 2010.

[Steichen2011a] Ben Steichen, Alexander O Connor, and Vincent Wade. Personalisation in the Wild Providing Personalisation across Semantic , Social and Open-Web Resources. In *HT'11: Proc. of the 22nd int. conf. on Hypertext and Hypermedia*, pages 73–82, 2011.

[Steichen2012b] Ben Steichen. *Adaptive Retrieval , Composition & Presentation of Closed-Corpus and Open-Corpus Information.* PhD thesis, Trinity College Dublin, 2012.

[Steichen2012c] Ben Steichen, Helen Ashman, and Vincent Wade. A comparative survey of Personalised Information Retrieval and Adaptive Hypermedia

techniques. *Information Processing & Management*, 48(4):698–724, July 2012.

[Stoyanov2009] Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. Conundrums in Noun Phrase Coreference Resolution : Making Sense of the State-of-the-Art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009.

[Strehl2003] A Strehl and J Ghosh. Cluster ensembles - A knowledge reuse framework for combining multiple partitions. *JMLR'03: The Journal of Machine Learning Research*, 3:583–617, 2003.

[Sugiyama2004] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive Web Search Based on User Profile Constructed without Any Effort from Users. In *WWW'04: Proceedings of the 13th international conference on World Wide Web*, pages 675–684, 2004.

[Sweetnam2011] M. S. Sweetnam and B. a. Fennell. Natural language processing and early-modern dirty data: applying IBM Languageware to the 1641 depositions. *Literary and Linguistic Computing*, 27(1):39–54, December 2011.

[Tai1979] Kuo-chung Tai. The Tree-to-Tree Correction Problem. *Journal of ACM*, (3):422–433, 1979.

[Tan2012] Ruyue Tan. Web Information Extraction Based on Visual Characteristics. *Information Technology Journal*, 2012.

[Tang2010] Jie Tang, Limin Yao, Duo Zhang, and Jing Zhang. A Combination Approach to Web User Profiling. In *KDD'10: ACM Transactions on Knowledge Discovery from Data*, volume V, pages 1–38, 2010.

[Tanudjaja2002] Francisco Tanudjaja and Lik Mui. Persona : A Contextualized and Personalized Web Search. In *HICSS '02: Proceedings of the 35th Annual*

Hawaii International Conference on System Sciences, volume 00, pages 1–9, 2002.

[Teevan2005] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR '05 Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, page 449, New York, New York, USA, 2005. ACM Press.

[Tsandilas2004] T. Tsandilas and M. C. Schraefel. Usable adaptive hypermedia systems. *New Review of Hypermedia and Multimedia*, 10(1):5–29, June 2004.

[W3c2004a] World Wide Web Consortium. RDF Primer, 2004.

[Weal2007] Mark J Weal, Harith Alani, Sanghee Kim, Paul H. Lewis, David E. Millard, Patrick a.S. Sinclair, David C. De Roure, and Nigel R. Shadbolt. Ontologies as facilitators for repurposing web documents. *International Journal of Human-Computer Studies*, 65(6):537–562, June 2007.

[Weissig2009] Yves Weissig and Thomas Gottron. Combinations of Content Extraction Algorithms. In *Workshop Information Retrieval*, pages 1–4, 2009.

[Weninge2008] Tim Weninge and William H. Hsu. Web Content Extraction Through Histogram Clustering. In *ANNIE'08: 18th International Conference on Artificial Neural Networks in Engineering*, pages 1–8, 2008.

[Weninger2008] Tim Weninger and William H. Hsu. Text Extraction from the Web via Text-to-Tag Ratio. In *DEXA'08: 19th International Conference on Database and Expert Systems Applications*, pages 23–28. Ieee, September 2008.

[Wexelblat1999] Alan Wexelblat and Pattie Maes. Footprints : History-Rich Tools for Information Foraging. In *CHI '99: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 1999.

[Xue2003] Nianwen Xue. Chinese Word Segmentation as Character Tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48, 2003.

[Yang2012] Haojin Yang, Christoph Oehlke, and Christoph Meinel. An Automated Analysis and Indexing Framework for Lecture Video Portal. In *ICWL'12: Proceedings of the 11th Int. Conf. on Web-based Learning, Sinaia, Romania, September 2-4, 2012*, pages 285–294, 2012.

[Ye2010] Nanhong Ye, Susan Gauch, Qiang Wang, and Hiep Luong. An Adaptive Ontology Based Hierarchical Browsing System for CiteSeerx. In *KSE'10: Second International Conference on Knowledge and Systems Engineering Knowledge and Systems Engineering*, pages 203–208. Ieee, October 2010.

[Yee2003] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *CHI '03 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 401, New York, New York, USA, 2003. ACM Press.

[Yeung2001] Ka Yee Yeung and Walter L Ruzzo. Details of the Adjusted Rand index and Clustering algorithms Supplement to the paper An empirical study on Principal Component Analysis for clustering gene expression data. *Bioinformatics*, pages 763–774, 2001.

[Yi2003] Lan Yi, Bing Liu, and Xiaoli Li. Eliminating Noisy Information in Web Pages for Data Mining. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296–305, 2003.

[Yi2003a] Lan Yi and Bing Liu. Web Page Cleaning for Web Mining through Feature Weighting. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, 2003.

[Yin2004] Xinyi Yin and Wee Sun Lee. Using link analysis to improve layout on mobile devices. In *WWW '04: Proceedings of the 13th conference on*

World Wide Web -, page 338, New York, New York, USA, 2004. ACM Press.

[Yin2005] Xinyi Yin and Wee Sun Lee. Understanding the function of web elements for mobile content delivery using random walk models. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, page 1150, New York, New York, USA, 2005. ACM Press.

[Zellweger1998a] Polle T. Zellweger, Bay-Wei Chang, and Jock D. Mackinlay. Fluid links for informed and incremental link transitions. In *HT'98: Proc. of Ninth ACM International Hypertext Conference*, pages 50–57, New York, New York, USA, 1998. ACM Press.

[Zhang2007] Yue Zhang and Stephen Clark. Chinese Segmentation with aWord-Based Perceptron Algorithm. In *ACL'07: Annual Meeting-Association for Computational Linguistics*, number June, pages 840–847, 2007.

[Zhang2010a] Kaixu Zhang, Maosong Sun, and Ping Xue. A Local Generative Model for Chinese Word Segmentation. *Information Retrieval Technology*, pages 420–431, 2010.

[Zhou2007a] Dong Zhou, Mark Truran, James Goulding, and Tim Brailsford. LLAMA: Automatic Hypertext Generation Utilizing Language Models. In *HT'07: Proc. of the 18th int. conf. on Hypertext and hypermedia*, pages 77–80, 2007.

[Zhou2008] Dong Zhou, Mark Truran, Tim Brailsford, Helen Ashman, and Amir Pourabdollah. LLAMA-B : Automatic Hyperlink Authoring in the Blogosphere. In *HT'08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 133–137, 2008.

[Zou2007] Jie Zou, Daniel Le, and George R Thoma. Structure and Content Analysis for HTML Medical Articles : A Hidden Markov Model

Approach. In *DocEng '07: Proceedings of the 2007 ACM symposium on Document engineering*, pages 5–7, 2007.

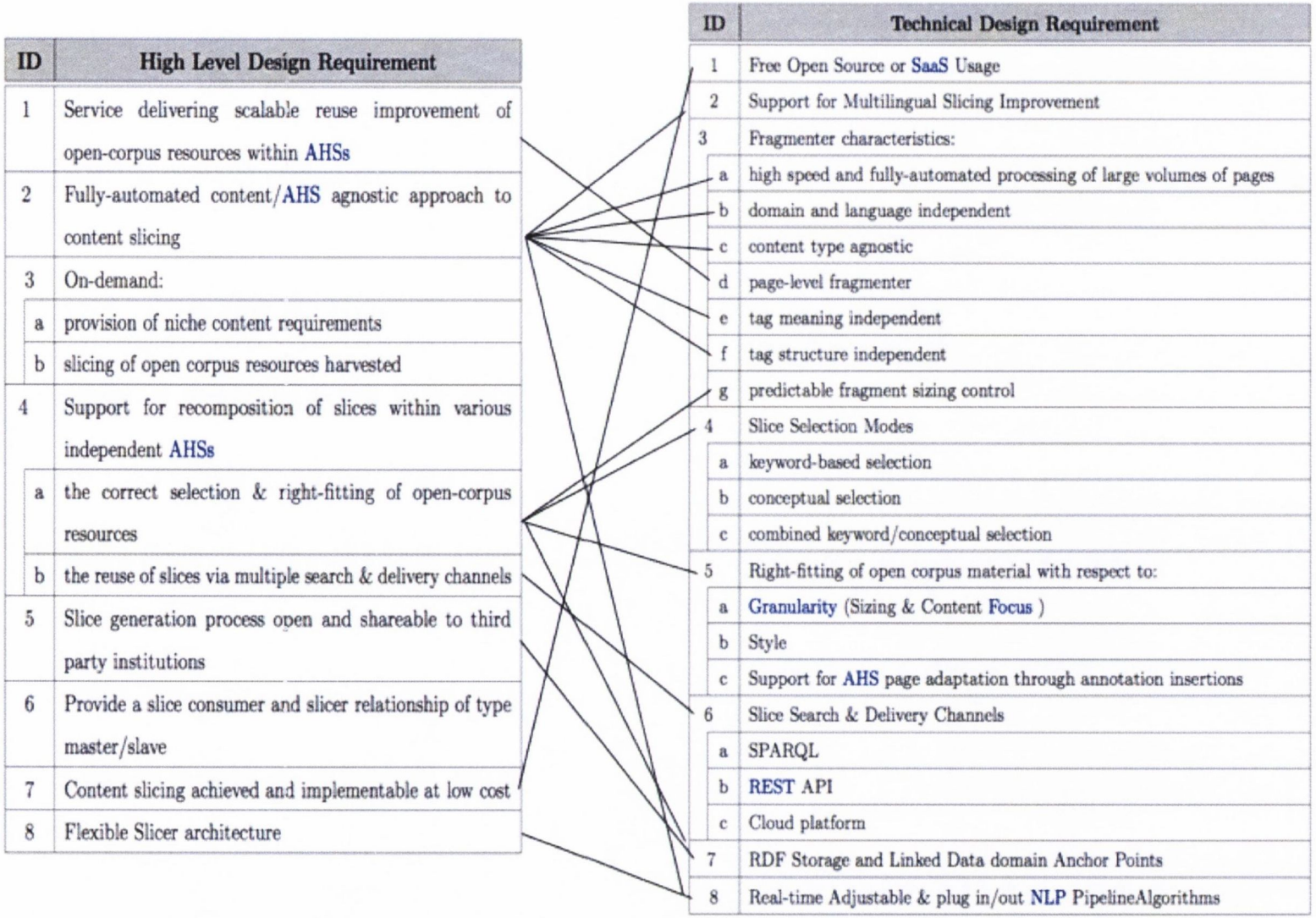
Appendix A

Slicepedia Design Requirement Dependencies

ID	Key Principle
1	**External Content Production**
a	Resources encapsulated as content packages, directly reusable within independent AHSs, delivered by a slicer service
b	AHS/slicer relationship of type master/slave
2	**Low Cost Automated Content Production**
3	**Open Corpus Content Reuse Improvement**
4	**Open Corpus Right-Fitting**
a	Correct selection and right-fitting of native open corpus resources to support recomposition within AHSs
b	Flexible slicer architecture
5	**On Demand Niche Right-Fitting**
a	AHS niche content requirements provided to slicer at run-time only
b	Selection and right-fitting of native resources based upon open domain models
6	**Content/AHS Agnostic Open-Corpus Reuse**
a	Content/AHS agnostic approach to open corpus reuse
b	Support for *search & delivery* of content packages over multiple channels
7	**Content Repurposing**
8	**Web of Slices**
a	Content packages generation process and underlying slicing resources open and shareable between independent institutions
b	Low cost slicer implementation

ID	High Level Design Requirement
1	Service delivering scalable reuse improvement of open-corpus resources within AHSs
2	Fully-automated content/AHS agnostic approach to content slicing
3	On-demand:
a	provision of niche content requirements
b	slicing of open corpus resources harvested
4	Support for recomposition of slices within various independent AHSs
a	the correct selection & right-fitting of open-corpus resources
b	the reuse of slices via multiple search & delivery channels
5	Slice generation process open and shareable to third party institutions
6	Provide a slice consumer and slicer relationship of type master/slave
7	Content slicing achieved and implementable at low cost
8	Flexible Slicer architecture

Figure A.1: High Level Design Requirement and KeyPrinciple Dependencies



ID	High Level Design Requirement
1	Service delivering scalable reuse improvement of open-corpus resources within AHSs
2	Fully-automated content/AHS agnostic approach to content slicing
3	On-demand:
a	provision of niche content requirements
b	slicing of open corpus resources harvested
4	Support for recomposition of slices within various independent AHSs
a	the correct selection & right-fitting of open-corpus resources
b	the reuse of slices via multiple search & delivery channels
5	Slice generation process open and shareable to third party institutions
6	Provide a slice consumer and slicer relationship of type master/slave
7	Content slicing achieved and implementable at low cost
8	Flexible Slicer architecture

ID	Technical Design Requirement
1	Free Open Source or SaaS Usage
2	Support for Multilingual Slicing Improvement
3	Fragmenter characteristics:
a	high speed and fully-automated processing of large volumes of pages
b	domain and language independent
c	content type agnostic
d	page-level fragmenter
e	tag meaning independent
f	tag structure independent
g	predictable fragment sizing control
4	Slice Selection Modes
a	keyword-based selection
b	conceptual selection
c	combined keyword/conceptual selection
5	Right-fitting of open corpus material with respect to:
a	Granularity (Sizing & Content Focus)
b	Style
c	Support for AHS page adaptation through annotation insertions
6	Slice Search & Delivery Channels
a	SPARQL
b	REST API
c	Cloud platform
7	RDF Storage and Linked Data domain Anchor Points
8	Real-time Adjustable & plug in/out NLP PipelineAlgorithms

Figure A.2: Design Requirement Dependencies