Using NLP Techniques to Enhance Content Discoverability and Reusability for Adaptive Systems

A thesis submitted to the

University of Dublin, Trinity College

in fulfilment of the requirements of the degree of

Doctor of Philosophy

Mostafa Bayomi

Knowledge and Data Engineering Group (KDEG)

School of Computer Science and Statistics

Trinity College, Dublin

Ireland

Supervised by Prof. Séamus Lawless

2019

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signature _____ Date _____

Mostafa Bayomi

Permission to lend or copy

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

Signature _____ Date _____

Mostafa Bayomi

Dedication

To my mother's soul

Acknowledgements

First and foremost, all thanks and praise are due to God, who has granted me this great success in my PhD, and has bestowed upon me all the necessary strength, health, wits, patience, and perseverance to complete it.

I would like to express my gratitude to numerous people who have helped me to complete my work over the last number of years. I would like to express my utmost gratitude to my supervisor Prof. Séamus Lawless for all his guidance, patience, feedback, encouragement, and for his great support personally and professionally throughout my PhD journey. I could not have done it without him.

I would like to thank all my colleagues in KDEG and ADAPT who have directly and indirectly supported me both personally and academically during the years of the PhD.

My utmost gratefulness and sincerest thanks to my father (Mohamed Bayomi), my brothers (Ahmed and Amr) and my sister (Rasha) who have always believed in me and for all their care, support and encouragement. To my lovely wife Alaa, for all her love, devotion, never-ending support and encouragement. To my kids, Hamza and Omar who are the joy of my life.

I would like to thank all my Egyptian friends in Dublin who have been a major part of this journey. And a special thanks to two very special people, Ramy Shosha and Rami Ghorab. Words of gratitude and thanks can never do them their rights.

And I've saved the best for last: My utmost gratefulness and sincerest thanks to my mother (Boushra Shams), whom I wish that she is with me in this moment. She has been and will always be the greatest supportive person in my life, especially in my PhD journey. You have left this world, but you will always be with me. Thanks mum for everything, words will never express my feelings for you. May God bless your soul.

Abstract

The volume of digital content resources written as text documents is growing every day, at an unprecedented rate. Because this content is generally not structured as easy-to-handle units, it can be very difficult for users to find information they are interested in, or to help them accomplish their tasks. This in turn has increased the need for producing tailored content that can be adapted to the needs of individual users. A key challenge for producing such tailored content lies in the ability to understand how this content is structured. Hence, the efficient analysis and understanding of unstructured text content has become increasingly important. This has led to the increasing use of Natural Language Processing (NLP) techniques to help with processing unstructured text documents. Amongst the different NLP techniques, Text Segmentation is specifically used to understand the structure of textual documents. However, current approaches to text segmentation are typically based upon using lexical and/or syntactic representation to build a structure from the unstructured text documents. However, the relationship between segments may be semantic, rather than lexical or syntactic.

Furthermore, text segmentation research has primarily focused on techniques that can be used to process text documents but not on how these techniques can be utilised to produce tailored content that can be adapted to the needs of individual users. In contrast, the field of Adaptive Systems has inherently focused on the challenges associated with dynamically adapting and delivering content to individual users. However, adaptive systems have primarily focused upon the techniques of adapting content, not on how to understand and structure this content. Even systems that have focused on structuring content are limited in that they rely upon the original structure of the content resource, which reflects the perspective of its author. Therefore, these systems are limited in that they do not deeply "understand" the structure of the content, which in turn, limits their capability to discover and supply appropriate content for use in defined contexts, and limits the content's amenability for reuse within various independent adaptive systems.

In order to utilise the strength of NLP techniques to overcome the challenges of understanding unstructured text content, this thesis investigates how NLP techniques can be utilised in order to enhance the supply of content to adaptive systems. Specifically, the contribution of this thesis is concerned with addressing the challenges associated with hierarchical text segmentation techniques, and with content discoverability and reusability for adaptive systems.

Firstly, this research proposes a novel hierarchical text segmentation approach, named C-HTS, that builds a structure from text documents based on the semantic representation of text. Semantic representation is a method that replaces keyword-based text representation with concept-based features, where the meaning of a piece of text is represented as a vector of knowledge concepts automatically extracted from massive human knowledge repositories such as Wikipedia. Using this approach, C-HTS represents the content of a document as a tree-like hierarchy. This way of structuring the document can be regarded as a hierarchically coherent tree that is useful for supporting a variety of search methods as it provides different levels of granularity for the underlying content. Secondly, this research proposes a novel content-supply service named CROCC. The aim of CROCC is to utilise the produced structure of C-HTS in order to overcome the limitations of the state of the art content-supply approaches. Finally, this research conducts an evaluation of the extent to which the CROCC service enhances content discoverability and reusability for adaptive systems.

Table of Contents

List of Figures

List of Tables

Acronyms

AHS	Adaptive Hypermedia System
APeLS	Adaptive Personalized eLearning Service
AR	Anaphora Resolution
DOM	Document Object Model
ESA	Explicit Semantic Analysis
HAPS	Hierarchical Affinity Propagation for Segmentation
IR	Information Retrieval
JSON	JavaScript Object Notation
LO	Learning Objects
LOM	Learning Object Metadata
LSA	Latent Semantic Analysis
MHTSS	Multi-granularity Hierarchical Topic-Based Segmentation System
NER	Named Entity Recognition
NLP	Natural Language Processing
PMCC	Personal Multilingual Customer Care
RDF	Resource Description Framework
SCORM	Sharable Content Object Reference Model
SME	Subject-Matter Expert
TFIDF	Term Frequencies - Inverse Document Frequency
UML	Unified Modelling Language
WSD	Word Sense Disambiguation
WWW	World Wide Web

1. Introduction

1.1 Motivation

A large proportion of digital content resources are written as text documents in the form of web pages, product manuals, news articles, research papers, etc. The volume of this content is growing at an unprecedented rate, making it very difficult for users to find information they are interested in or to help them accomplish their tasks (Uchyigit, 2009). The reason is that these resources are generally not properly structured into easy to handle units. Hence, efficient analysis and understanding of unstructured text content is becoming increasingly important (Alani et al., 2003). This has led to the increasing use of Natural Language Processing (NLP[1]) techniques to help with processing unstructured text documents (Beck et al., 2014; Sathiyamurthy and Geetha, 2011). The fundamental objective of 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 (Collobert et al., 2011). In the real world, natural language text usually appears as sequential patterns without explicitly defined boundaries to identify how the text is structured. Amongst the various NLP techniques that have been developed, Text Segmentation is used to identify boundaries in natural language text, and hence understand the structure of textual documents (Badjatiya et al., 2018; Wang et al., 2017; Eisenstein, 2009; Hearst, 1997). Current approaches to text segmentation are based upon using lexical and/or syntactic representation to identify the coherent segments of text (Azzopardi et al., 2017; Kazantseva and Szpakowicz, 2014). However, the relationship between segments may be semantic, rather than lexical or syntactic. Furthermore, text segmentation research has mainly focused on techniques that can be used to process text documents but not on how these techniques can be utilised to produce tailored content that can be adapted to the needs of individual users.

The field of content adaptation has aimed to assist users with the problem of information overload, and focused on the challenges associated with the growing body of digital content and methods to dynamically adapt and deliver it to individual users (Janati et al., 2018; Bunt et al., 2007). One area of content adaptation is Adaptive Systems[2]. One of the

[1] NLP also refers to *Neuro-Linguistic Programming*. In this research, it refers to Natural Language Processing.
[2] Adaptive Systems are also commonly referred to within the research community as Adaptive Web Systems or Adaptive Hypermedia Systems (AHSs). In this thesis, "adaptive systems" means any system that tailors content to user's needs.

main services that adaptive systems offer to their users is the provision of content[3] that is tailored to individual users' needs. In order to provide such content, adaptive systems utilise different techniques to incorporate content that meets the requirements of their users.

Early adaptive systems primarily focused on content adaptation techniques rather than the processing and production of the content itself (Dieberger & Guzdial, 2003; Conlan & Wade, 2004; De Bra et al., 2003; Brusilovsky, 2004). Such systems have traditionally relied upon the manual processing and production of content (Dieberger & Guzdial, 2003). This manual processing makes the resources available in these systems highly curated, and easily discovered and adapted to the user's preferences. However, the result, typically, is the production of relatively low volumes of content at high cost which makes these systems only able to satisfy a narrow range of content requests (Levacher, 2014). Furthermore, the labour-intensive requirements imposed upon the manual processing of content resources results in a limited capability of content reusability within different systems.

As a result, various adaptive systems have been proposed to address these challenges by employing automatic techniques to incorporate and process content. They have mainly focused on leveraging and utilising open corpus resources available on the World Wide Web (Heufemann et al., 2013; Sosnovsky et al., 2012; Lawless, 2009; Weal et al., 2007). This in turn allowed these systems to incorporate a wider range of open corpus content resources that cover a more diverse range of information needs (Smith & Blandford, 2003). However, since these approaches have prioritised the development of automatic techniques to support the use of open corpus content, these techniques typically used the content resources in the form they were created, as static one-size-fits-all content objects, with limited control over content granularity. As pointed by (Bunt et al., 2007), presenting the incorporated content resources in their native form allows more content to be "*visible to the user. However, the more content is shown, the higher the chance of generating information overload and reducing attention to the most relevant information, defeating one of the very reasons for having adaptive systems in the first place*".

Additionally, when content reuse has been achieved in these systems, it is traditionally performed manually (Henze and Nejdl, 2001) or at best using automated approaches that

[3] Content has different types such as: textual content, and multimedia content (image, video, audio, or animation) among others. The focus of this research is on textual content. Hence, in this thesis, the term content refers to textual content.

treat such resources as document level packages only. As pointed out by (Lawless, 2009), "*there is an inverse relationship between the potential reusability of [...] content and its granularity*". The reuse of open web resources in their native form could be improved if reused in different sizes.

As a result, other systems tried to overcome these problems and focused on performing adaptation at a finer level of granularity (Levacher et al., 2012b). In these approaches, the harvested resources are processed and structured into coherent fragments. ArtEquAKT (Millard et al., 2003; Weal et al., 2007) for example, utilised information extraction and knowledge management techniques to automatically extract parts of content resources (paragraphs) to create dynamic biographies of artists from content available on the web. Another example is the PMCC[4] system (Steichen, 2012) that delivers personalised content to individual users using open corpus content as fragments of text. The harvested content is fragmented based on its HTML structure using a wrapper-based content fragmentation approach (Bunt et al., 2007) to identify regions of pages in order to produce individual fragments of content. Slicepedia (Levacher, 2014) was introduced as a web service to process open corpus resources and extract content for reuse by right-fitting it to the specific content requirements of individual content consuming applications. Slicepedia fragments the harvested open corpus content into segments based on their HTML structure.

Although these systems have demonstrated their ability to automatically process textual content and hence enhance its discoverability and reusability, they are limited in that they rely only upon the original structure of the content resource that reflects the needs and perspective of its author. While each adaptive system has its own content requirements (based on its users), relying upon such structure does not reflect these requirements. Furthermore, content resources that do not possess any layout structure (e.g. plain text) or do not make use of orthographic information (e.g. content is not structured as paragraphs), cannot be effectively processed and reasoned about by these systems. Additionally, these systems are limited in that they do not deeply "understand" the structure of the content, which, in turn, limits their capability to supply appropriate content for use in defined contexts. Understanding the structure of content requires a deep understanding of the meaning of that content.

[4] Personal Multilingual Customer Care

In order to utilise the strength of NLP techniques and to overcome the challenges of understanding unstructured text content, this thesis investigates how NLP techniques can be utilised in order to enhance the supply of content to adaptive systems. Specifically, this thesis focuses on the use of text segmentation, as a technique for structuring textual documents, to enhance content discoverability and reusability for adaptive systems.

In this thesis, two novel hierarchical text segmentation approaches are presented. The two approaches are: ***OntoSeg*** (Bayomi et al., 2015) and ***C-HTS*** (Bayomi & Lawless, 2018). Both approaches use the semantic representation of textual content in order to segment it and produce a tree-like hierarchical structure. This way of building the document structure can be regarded as a hierarchically coherent tree that is useful to support a variety of search methods as it provides different levels of granularity for the underlying content. This tree is then traversed to obtain different levels of content granularity that facilitate content discoverability and reusability. Each approach was evaluated independently to explore its efficiency in performing its task. Additionally, a novel content-supply service called CROCC (Customised Reuse of Open- and Closed-corpus Content) is presented. The aim of CROCC is to utilise the produced structure of the segmentation algorithm (C-HTS[5]) in order to overcome the limitations of the state of the art content-supply approaches. Furthermore, an evaluation of the extent to which the CROCC service enhances content discoverability and reusability for adaptive systems is presented.

In this thesis, CROCC (using C-HTS) is designed and evaluated with reference to the supply of content to adaptive systems. However, the work presented in this thesis can potentially be used by a range of different applications that rely on structuring and understanding content such as Recommender Systems (Karimi et al., 2018) and Passage Retrieval systems (Cohen and Croft, 2018).

Adaptive systems rely on different models to produce adaptive content compositions according to their users' needs. It is not within the scope of this research to design and build an adaptive system. Rather, the aim of this thesis is to propose novel approaches for structuring textual content based on its semantic representation and to utilise the produced structure in a content-supply service to enhance the discovery and reuse of content for adaptive systems.

[5] Evidence from experiments demonstrated that C-HTS is performing better than OntoSeg (Chapter 4). Hence, C-HTS is employed in CROCC to build the structure for content resources (as will be discussed in Chapter 5).

1.2 Research Question

The question that this research seeks to answer is:

To what extent can the semantic representation of unstructured textual content be exploited by novel text segmentation approaches to build a document structure?

To assess whether the structure produced by the proposed approaches is of benefit to content adaptation, a further question will be addressed:

Can the produced structure be used to enhance the discoverability and reusability of content for adaptive systems?

In light of these research questions, and the research objectives outlined in section 1.2.1 below, this research aims to propose novel approaches for producing a structure out of textual content in order to improve content discoverability and reusability for adaptive systems.

1.2.1 Research Objectives

In order to address the research questions outlined above, the following research objectives were identified for this thesis:

RO 1: Perform a state of the art survey on NLP techniques, specifically text segmentation as a technique for structuring textual content. The aim of this survey is to investigate how text segmentation is used to analyse and understand text to produce a structure from unstructured textual documents. Additionally, perform a state of the art survey on adaptive systems as content adaptation applications, to investigate how they process content and the different techniques they utilise in order to facilitate the discovery and reuse of this content. The survey should also review how state of the art adaptive systems utilise NLP techniques in order to provide adaptive content.

RO 2: Examine the different methods and techniques that can be used to enhance the performance of text segmentation using the semantic representation of text, and develop a new text segmentation approach to enhance the understanding of unstructured textual documents. This also involves the evaluation of the effectiveness of the proposed approach in processing and structuring content.

RO 3: This PhD research takes adaptive systems as the target application scenario. To enhance the content discoverability and reusability, it is important to understand

the structure of that content. The proposed hierarchical text segmentation approach makes it possible to build a structure out of content resources based on the semantic representation of text. In this context, a new content-supply service that utilises the structure produced by the proposed segmentation approach needs to be built. The design of this service should be focused on exploiting the produced structure in order to overcome the limitations of the state of the art content-supply approaches.

RO 4: Evaluate the extent to which the proposed content-supply service can enhance the discovery and reuse of content for adaptive systems.

1.3 Research Contributions

This work makes notable contributions to the state of the art of unstructured textual content analysis and understanding, along with content adaptation. These contributions are illustrated throughout this thesis. The **major contribution** of this research is the use of NLP techniques, specifically text segmentation, to analyse and build a structure from text documents based on the semantic representation of text. This structure is utilised by a novel content-supply service in order to enhance content discoverability and reusability for adaptive systems. To build a structure from text documents, this research proposes two novel hierarchical text segmentation algorithms based on the semantic representation of text, OntoSeg (Chapter 3) and *C-HTS* (Chapter 4).

OntoSeg (**Onto**logical **Seg**mentation) (Bayomi et al., 2015) is a novel approach to text segmentation that uses the *semantic similarity* between text blocks based on an ontology, and uses a Hierarchical Agglomerative Clustering (HAC) algorithm to represent the text as a tree-like hierarchy that is semantically structured.

Evidence from experiments conducted as part of this research indicates that although OntoSeg is able to produce a hierarchical structure out of text based on its semantic representation, it did not perform well against the state of the art approaches and thus, its performance needs to be enhanced through improved understandability of text by exploring the semantic relatedness between text blocks rather than just using the semantic similarity. As argued by (Budanitsky and Hirst, 2006), *relatedness* is more general than *similarity* since dissimilar entities may also be semantically related by other relationships such as meronymy, antonymy, functional relationship or frequent association.

As a result, another algorithm called C-HTS (**C**oncept-based **H**ierarchical **T**ext **S**egmentation) (Bayomi and Lawless, 2018) is presented. C-HTS is a hierarchical text segmentation approach that uses the explicit semantic representation of text to measure the *semantic relatedness* between text blocks. The semantic representation of text is a method that replaces keyword-based text representation with concept-based features, automatically extracted from massive human knowledge repositories such as Wikipedia. C-HTS represents the meaning of a piece of text as a weighted vector of *knowledge concepts*, in order to reason about text. Similar to OntoSeg, C-HTS produces the content of a single document as a tree-like hierarchy. This way of structuring the document can be regarded as a hierarchically coherent tree that is useful to support a variety of search methods as it provides different levels of granularity for the underlying content.

This thesis also proposes a novel content-supply service named CROCC (**C**ustomised **R**euse of **O**pen- and **C**losed-corpus **C**ontent) that utilises the structure produced by C-HTS in order to overcome the limitations of the state of the art content-supply approaches. CROCC is a service which harvests content resources from open and closed corpus in their native form and builds a structure out of each content resource without the reliance upon its original structure. The service builds the structure of a content resource based on its semantic representation (using C-HTS) and delivers content slices[6] according to the needs and requirements of individual adaptive systems. The thesis also presents a task-based experiment to evaluate the extent to which the CROCC service can enhance the discovery and reuse of content for adaptive systems.

A **minor contribution** of this research is a concept space that was built from a Wikipedia snapshot (April 2017) to be used for the explicit semantic analysis of text within C-HTS. This concept space is publicly available[7]. Another **minor contribution** is the implementations of the two hierarchical text segmentation algorithms proposed in this thesis, OntoSeg and C-HTS. Implementations of both algorithms have been open-sourced and made publicly available[8,9].

To date, three research papers directly related to this research have been published:

[6] A slice is a piece of content (one or more sentences) that originates from pre-existing content resource
[7] https://goo.gl/JZhEvm
[8] https://github.com/bayomim/OntoSeg
[9] https://github.com/bayomim/C-HTS

1. Bayomi, M., Levacher K., Ghorab, M.R., and Lawless, S. "*OntoSeg: A Novel Approach to Text Segmentation using Ontological Similarity*". In the proceedings of the 5th ICDM Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction, ICDM SENTIRE. Held in conjunction with the IEEE International Conference on Data Mining, ICDM 2015. Nov 14th, 2015. Atlantic City, NJ, USA.

This publication describes the OntoSeg algorithm that uses the semantic similarity between text blocks. The publication also describes the experiments that have been carried out in order to evaluate the performance of OntoSeg in comparison with state of the art text segmentation approaches.

2. Bayomi, M. & Lawless, S. "*C-HTS: A Concept-based Hierarchical Text Segmentation approach*". In the Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). Miyazaki, Japan: European Language Resources Association (ELRA).

This publication describes the C-HTS algorithm that uses the semantic relatedness between text blocks. The publication describes the approach used by C-HTS to apply hierarchical text segmentation and the concept space that has been built from Wikipedia in order to measure the semantic relatedness between text blocks. The publication also describes the experiments that have been carried out in order to evaluate the performance of C-HTS against the state of the art hierarchical text segmentation approaches.

3. Bayomi, M. "*A Framework to Provide Customized Reuse of Open Corpus Content for Adaptive Systems*." In the Proceedings of the 26th ACM Conference on Hypertext & Social Media. HT '15. Northern Cyprus, pp 315–318. ACM, 2015.

This publication describes a preliminary version of the CROCC framework that has been built to facilitate the use of text segmentation in a content adaptation scenario.

Another publication describing the CROCC service and its evaluation (detailed in Chapter 5 and Chapter 6) is underway and will target the ACM Hypertext conference[10].

1.4 Research Methodology

This research addressed the abovementioned research questions and objectives by following a number of iterative steps that involved theoretical investigation, experimental

[10] https://human.iisys.de/ht2019/

design, technical implementation, and quantitative and qualitative evaluation. This section provides an overview of the research methodology followed.

Initially, a state of the art survey on NLP techniques in general and a focused review on text segmentation was undertaken. The review investigated the different text segmentation approaches used for analysing and structuring textual documents and identified points of strength and weakness in these approaches and how they could be enhanced. The survey then included a review of adaptive systems, their anatomy, models and the different content models that adaptive systems rely on. The survey also investigated the different approaches for content discoverability and reusability used within adaptive systems. The survey then investigated the different NLP techniques used in adaptive systems in order to structure textual content (RO 1).

Based on the influences derived from this survey, two new text segmentation algorithms that use the semantic representation of content were proposed. The two algorithms, namely OntoSeg (Chapter 3) and C-HTS (Chapter 4), use the semantic representation of content to reason about it. Both algorithms were proposed to enhance text understandability and hence build a semantic structure to change the static and inflexible nature of textual content. Both algorithms were evaluated independently, using a set of experiments, to assess their performance in text segmentation tasks (RO 2).

As discussed in section 1.1, text segmentation research has mainly focused on techniques that can be used to process text documents but not on how these techniques can be utilised to produce tailored content that can be adapted to the needs of individual users. Thus, in order to utilise the C-HTS algorithm in content adaptation, a content-supply service named CROCC was developed (Chapter 5). CROCC was offered as an intelligent content-supply framework designed based on the influences derived from the state of the art review on adaptive systems. The fundamental objective of CROCC is to utilise the structure produced by C-HTS to enhance content discoverability and reusability for adaptive systems (RO 3).

To evaluate the extent to which the CROCC service enhances the discovery and reuse of content for adaptive systems, a task-based experiment was carried out (RO 4). Since designing and building an adaptive system is not within the scope of this thesis, this experiment did not focus on evaluating the process of content use within an actual adaptive system. Rather, the experiment focused on evaluating the content-supply mechanism of

CROCC and the quality of the slices produced by the service, according to the specific requirements of a set of content requests that could be sent by an adaptive system.

1.5 Thesis Overview

The remainder of this thesis is organised as follows:

Chapter 1 presents a state of the art survey of NLP and adaptive systems. The survey begins with a comprehensive review of a number of different NLP techniques. The field of text segmentation is reviewed, through an overview of the categories of text segmentation proposed so far in the literature, along with different techniques for segmentation and their strengths and limitations. The survey then focuses on hierarchical text segmentation and investigates how hierarchical text segmentation is used to analyse text to produce a structure from unstructured textual documents. The chapter then provides an overview of adaptive systems and describes their anatomy. Their strengths and limitations are also presented. Types of content such as closed and open corpus content are then outlined. Following this is a review of how state of the art adaptive systems utilise the different techniques in order to discover content that is suitable for their users' needs. Reusability techniques applied by adaptive systems are then reviewed. The chapter also comprises a review of a range of NLP techniques utilised by adaptive systems in order to structure textual content.

The following chapters outline the design of the various elements of research presented in this thesis. They build on the influences from the state of the art survey and outline the design of the theoretical approaches proposed to meet the research questions and objectives of this thesis.

Chapter 3 presents the design of OntoSeg, a new algorithm for text segmentation that hierarchically represents the conceptual structure of content based on the semantic representation of text. The chapter also describes a set of experiments that have been carried out in order to evaluate the performance of OntoSeg using well-known evaluation metrics. The OntoSeg evaluation comprises a set of experiments where each experiment evaluates OntoSeg from a different perspective. The performance of OntoSeg is also compared against a set of state of the art approaches using a dataset widely used in the literature.

Chapter 4 presents the design of C-HTS, the hierarchical text segmentation approach that focuses on the semantic relatedness between text constituents. The technical aspects

related to building a concept space from Wikipedia are presented. Wikipedia was selected in this research as it is considered the largest and fastest growing knowledge base in existence as it is a collaborative effort that combines the knowledge of hundreds of thousands of people. The chapter also describes a set of experiments that have been carried out in order to evaluate the performance of C-HTS using two different datasets in the hierarchical text segmentation literature. Also, to assess the efficiency of C-HTS, its performance is compared against state of the art hierarchical text segmentation approaches. The chapter then describes two sets of experiments that have been carried out in order to: (1) validate the efficacy of using Wikipedia as the underlying knowledge base for the semantic representation of text in C-HTS, and (2) validate the efficacy of using the semantic representation of text rather than its lexical representation.

Since C-HTS uses Wikipedia as the underlying knowledge base to reason about text, and since the amount of knowledge in Wikipedia is expanding, such expansion, and the growth of information available in the knowledge base should impact the accuracy of the segmentation process. Hence, this chapter presents an evaluation of how this knowledge expansion impacts upon the segmentation accuracy of C-HTS.

Chapter 5 presents the CROCC service and describes the different modules in the service. The chapter starts by presenting the influences derived from the state of the art review and the key principles that impacted the core properties of the CROCC service. The chapter then presents the design aspects of CROCC service along with an explanation of how each component in the service influences the content provision process. After that, the chapter discusses how the design of the service adheres to the key principals derived from the state of the art influences. Details of a prototype implementation of the service are then presented.

Chapter 6 describes the experiment that has been carried out in order to evaluate the extent to which the CROCC service can enhance the discovery and reuse of content for adaptive systems. The chapter starts by describing the methodology applied in this experiment. After that, the chapter outlines how content resources from closed and open corpus were acquired for the purpose of the experiment. A baseline system is then introduced, which has been developed to compare its performance against the CROCC service. The experimental setup and the evaluation system that has been built for this experiment are

then described. The chapter concludes by presenting the analyses carried out, along with the findings derived from this analysis.

Chapter 7 concludes the thesis with a summary of the key contributions of this research, a discussion of how well the objectives were met and how the research questions were answered, and a discussion of future work that may be carried forward from this thesis.

2. State of the Art

2.1 Introduction

This chapter presents a state of the art survey of NLP techniques, specifically text segmentation, and presents a review of adaptive systems. 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. The aim is also to contribute to the first objective of this thesis (RO 1) by reviewing the field of text segmentation to investigate how text segmentation is used to analyse and understand unstructured textual documents. The survey reviews a number of different approaches to text segmentation along with their limitations. The survey also reviews adaptive systems, as content adaptation applications, to get an insight into how they process content to adapt it to their users' needs. Additionally, the survey reviews content discovery and reuse techniques within adaptive systems. These techniques and systems are reviewed in terms of their strengths and their limitations. The survey also reviews a range of different NLP techniques utilised by adaptive systems.

2.2 Natural Language Processing

2.2.1 Overview

Natural Language Processing (NLP) is a phrase used to describe a range of computational techniques, based upon linguistic theory, for the automatic analysis and representation of natural language (Cambria & White, 2014). 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 (Collobert et al., 2011). During the last decades, scientific efforts and the increasing availability of computational resources have made it possible to come closer to the goal of understanding textual content (Riedl 2016). Existing NLP techniques have been applied successfully in a wide range of areas such as Machine Translation (McCann et al., 2017), Information Extraction (Stanovsky et al., 2018; Fader et al., 2011), and Information Retrieval (Masumura et al., 2017; Mitra and Craswell, 2017) among others.

In 1950, Alan Turing presented the Turing Test (Turing, 1950). The test consists of a text conversation between two participants. One participant is a human and the other is a computer. The aim of this test is to see if machine can think. Hence, Turing proposed the test

as a game, in which a computer's use of language would form the basis for determining if the machine could think. Turing defined a machine as intelligent if the evaluator cannot distinguish the machine from the human. Following that, in 1954, John Hutchins proposed the Georgetown-IBM system that involved the translation of Russian sentences into English (Hutchins, 2005). Similar to Turing system, (Weizenbaum, 1966) proposed ELIZA program 1966. ELIZA was an early natural language processing system capable of carrying on a limited form of conversation with a user by using pattern matching to process the input and translate it into suitable outputs.

Over time, a vast amount of NLP approaches have been drawn from these systems (Jurafsky, 2000). NLP systems usually 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 particular problem in processing natural language. These tasks are usually subdivided into two broad classes based upon whether they consist of low-level or high-level processing tasks (Levacher, 2014).

2.2.2 Low-level NLP Tasks

Low-level NLP tasks are usually used as a pre-processing step for other NLP tasks. An example of a low-level task is ***Tokenisation***, which is also referred as word segmentation (Mullen et al., 2018; Goldwater et al., 2006). This task aims at handling word structure by separating a stream of text into a consecutive set of tokens which roughly correspond to "words" (Chang & Manning, 2014). Tokenisation mainly depends on word boundaries, such as space, to identify the different tokens in the given piece of text. 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, Asian languages such as Chinese and Japanese have no explicit word boundaries, which make tokenisation a challenging task (Zhang et al., 2010). Even in western languages, valid words are often not identical to space-separated tokens. For example, proper nouns such as "United Kingdom" or idiomatic phrases such as "with respect to" actually function as a single word (Mochihashi et al., 2009). Nevertheless, different approaches have been proposed to tackle these problems (Manning et al., 2014).

On the other hand, the task of ***sentence splitting*** is concerned with segmenting text into sentences (Xu et al., 2017; Mikheev, 2000). While tokenisation relies on the delimiters between tokens, sentence splitting depends on the boundaries between sentences in raw text. Many NLP tasks require their input to be divided into sentences. For instance, to

summarise text each sentence needs to be identified in order to measure how important this sentence is to be included in the final summary (Bayomi et al., 2016; Mihalcea & Tarau, 2004). In a text segmentation task, to identify coherent segments in the given text, there is a need to split it into sentences and then measure how these sentences are similar (or related) to each other (Bayomi & Lawless, 2018; Bayomi et al., 2015; Choi, 2000). Both tokenisation and sentence splitting tasks are generally used as the first processing tasks applied to a raw natural language text, which directly influence the results of the subsequent tasks.

While some NLP tasks might only require text to be tokenised or segmented into sentences, some other tasks require various tokens to be grouped together based upon their common root in the language. For example, in a search engine (Information Retrieval task), we want to find relevant results not only for the exact word we typed in the search bar, but also for the other possible forms of the words we used. If the typed word is "skirts", for example, it is very likely we will want to see results containing the word "skirt". Hence, grounding words into their root is an essential task. This task is usually referred to ***Lemmatisation*** (Chakrabarty et al., 2017; Joel et al., 2004) or ***Stemming*** (Hajeer et al., 2017; Willett, 2006). Both tasks aim to reduce the inflectional forms of each word into a common base or root. For example, words such as, play, plays, played, playing all possess the same root of "play". However, both tasks are different in the way they work and therefore so is the result that each of them returns. Essentially, stemming algorithms cut off the end or the beginning of the word based on a list of common prefixes and suffixes that can be found in a word. This makes such algorithms limited in some cases. For example, for the two words "Studies" and "Studying", although their root form is "Study", a stemming algorithm would reduce the two words, based on their suffixes, into "Studi" and "Study" respectively. On the other hand, lemmatisation algorithms usually use a vocabulary and morphological analysis of words to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma*. For the previous example, a lemmatisation algorithm can use a detailed dictionary to look through and link the two words back to their lemma, "Study".

Another task that is typically used as a pre-processing task is the **Part-Of-Speech (POS)** tagging (Farrah et al., 2018; Stratos et al., 2016). POS tagging explains how a word is used in a sentence by labelling each token with a unique tag that indicates its syntactic function, such as, noun, pronoun, verb, adverb, etc. (Santos & Zadrozny 2014). In a given text, many words, especially common ones, can serve as multiple parts of speech. For

example, “Play” can be verb (I play football every week) or can be a noun (I watched the play and it was wonderful). Different taggers have been proposed for the English language. (Shen et al., 2007) for example, propose a sequence classification approach for the English language that obtained an error rate of 2.67% on standard benchmarks. On the other hand, different taggers have been proposed for other languages (Habash & Rambow 2005). (Stratos et al., 2016) proposed an unsupervised part-of-speech (POS) tagger by learning Hidden Markov Models (HMMs), which they call anchor HMMs, where they extend the non-negative matrix factorization framework proposed by (Arora et al., 2013). Various POS taggers have been proposed for languages other than English. (Farrah et al., 2018), for example, proposed a rule-based hybrid tagger for Arabic language that uses an artificial neural network classifier to determine the appropriate tags of an Arabic text. The first step in their approach is to use the affixes in text to understand the nature of the word and its tags according to grammatical rules. The second step then is to apply a transliterated mechanism on text to convert it into Roman letters. This transliterated text is then used as an input of the classifier based on the neural networks. After that, the output of the two steps is used to identify the tag of each word.

2.2.3 High-level NLP Tasks

This category of tasks usually rely on the output from the pre-processing tasks presented in the previous section. One of these tasks is ***Information Extraction (IE)*** that is concerned with extracting semantic information from text (Li et al., 2018; Chang et al., 2006). The aim of IE algorithms is to extract structured information from unstructured documents. As stated by (Cowie & Lehnert, 1996): “*Information Extraction (IE) is the name given to any process which selectively structures and combines data which is found, explicitly stated or implied, in one or more texts. The final output of the extraction process varies; in every case, however, it can be transformed so as to populate some type of database.*” Hence, the field of Information Extraction essentially comprises different (sub) tasks, among them, which are of particular relevance for this thesis, is Named Entity Recognition (NER) (Gabbard et al., 2018).

Named Entity Recognition (NER) algorithms aim to detect entities within text and assign a type for each found entity, such as person, location, organization, etc. NER is essential to recognise information units like names, including person, organization and location names, and numeric expressions including time, date, money and percent expressions (Yadav and Bethard, 2018; Nadeau and Sekine, 2007). These unites are called named

entities which carry key information in a sentence and serve as important targets for most language processing systems (Mohit, 2014). Early NER approaches relied upon handcrafted rule-based algorithms and lexicons (Nadeau and Sekine, 2007). As the task evolved into a statistical learning problem, modern approaches have moved towards the use of machine learning techniques (Yadav and Bethard, 2018). (Agerri and Rigau, 2016) for example, proposed a multilingual NER approach (called *ixa-pipe-nerc*[1]) which learns supervised models via the Perceptron algorithm (Collins, 2002). (Habibi et al., 2017) presented a NER approach for biomedical text using long short-term memory network-conditional random field (LSTM-CRF). Their approach combines deep learning and word embeddings techniques and evaluation results showed that their approach outperformed other NER tools that do not use deep learning or use deep learning methods

Another high-level task in the NLP field consists of the ***Word Sense Disambiguation (WSD)***. WSD is the task of identifying the meaning of a term, when the term has multiple meanings, based upon the context of where it appears (Raganato et al., 2017; Agirre et al., 2014). For example, "light" can mean "not heavy" or "illumination", what identifies its meaning is the context of where "light" is used. While most of the time humans do not even think about the ambiguities of language, machines need to process unstructured textual content to understand it and reason about it. WSD algorithms mainly rely on external knowledge resources to associate the most appropriate senses with words in context (Agirre & Stevenson 2006). Examples of these knowledge resources are: Thesauri (Cañas et al., 2003), Machine Readable Dictionaries (Basile et al., 2014; Lesk, 1986) and Ontologies (Philpot et al., 2005). For example, (Banerjee & Pedersen 2002) proposed an adaptation of the Lesk algorithm (Lesk 1986) for word sense disambiguation. While the Lesk algorithm relies upon a standard dictionary in order to find overlaps between neighbouring words, they extended the algorithm and used the lexical database WordNet (Miller 1995) as the source of glosses for their approach.

Another NLP task is ***Automatic Text Summarisation***. Text summarisation is the process of abstracting key content from information sources. The goal of automatic summarisation is to process the source text to produce a shorter version of the information contained in it then present this version in a way that suits the needs of a particular user or application (Bayomi et al., 2016). Various techniques have been proposed in the literature for

[1] https://github.com/ixa-ehu/ixa-pipe-nerc

the automatic summarisation of text, some of which are supervised, while others are unsupervised. Supervised techniques involve the need for an existing dataset of example summaries (Cruz et al., 2006). In contrast, unsupervised techniques do not rely upon any external knowledge sources, models or on linguistic processing and interpretation to summarise text (Mihalcea & Tarau 2004). There are two primary approaches to automatic summarisation. *Extractive* methods work by selecting a subset of existing words, phrases, or sentences from the original text to form the summary (Vodolazova et al., 2013). In contrast, *Abstractive* methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might generate (Fiszman & Rindflesch 2003). Such a summary may contain words that are not included in the original text.

2.2.4 Summary

This section presented a general overview of NLP algorithms including the most common low-level and high-level tasks typically involved in this area. However, NLP field has many other tasks, each of these tasks represent a research field of its own and attempts to solve a particular problem in processing natural language. Example of other NLP tasks, among others, are: Machine Translation (Moussallem et al., 2018), Sentiment Analysis (Verma and Thakur, 2018; Medhat et al., 2014), Relationship Extraction (Zeng et al., 2015) and Speech recognition (Zhang et al., 2017; Povey et al., 2011).

It is worth mentioning that NLP tasks are not mutually exclusive. For example, for a Relationship Extraction (RE) task, an NER system should be used to extract entities for which the RE algorithm can identify the relationships among them (e.g. who is married to whom). Additionally, these tasks are very closely intertwined.

2.3 Text Segmentation

2.3.1 Overview

A large proportion of digital content resources are written as text documents in the form of web pages, product manuals, papers, etc. The volume of this content is growing at an unprecedented rate, making it very difficult for users to find interesting information or information which helps the user complete their task (Uchyigit, 2009). Several research fields have emerged which focus on the challenges associated with this growing body of content, and the methods used to understand it, in order to find information stored in

unstructured text documents. One of these research fields is Text Segmentation (Pak and Teh, 2018; Tsunoo et al., 2017; Glavaš et al., 2016; Choi, 2000; Hearst, 1997).

Text segmentation is the process of placing boundaries within text to create segments according to some task-dependent criterion. It aims to divide text into coherent segments which reflect the sub-topic structure of the text. Text segmentation algorithms are widely used as an essential step for Information Retrieval (Prince and Labadié, 2007; Llopis et al., 2002) and several NLP tasks such as text summarisation (Bokaei et al., 2016; Boguraev and Neff, 2000), Question Answering (Riahi et al., 2012) and automatic generation of eLearning Courses (Beck et al., 2014). In Information Retrieval, a document is segmented into distinct topics and only the topical segments relevant to the user's needs are retrieved. Segmentation not only provides more accurate information to the user, but also reduces the burden on the user of having to read the whole document. In document summarisation, a document is segmented into topics and then each topic is summarized independently. This process guarantees that the final summary covers all the key topics in the document.

Various synonyms in the literature are used to refer to text segmentation such as: Linear Text Segmentation (Badjatiya et al., 2018), Hierarchical Text Segmentation (Tsunoo et al., 2017), Topic Segmentation (Wang et al., 2017), Text Boundaries or Boundary Determination (Jamil et al., 2015; Labadié and Prince, 2008), and Topic Boundaries (Kim and Cho, 2014). Furthermore, text segmentation has been categorised from a number of different points of view. The following subsections present a number of these categorisations.

2.3.2 Content-based and Discourse-based

Content-based approaches focus on the story content and resolve the segmentation problem by relying on some measure of the difference in word usage on the two sides of a potential boundary: the larger the difference, the more indicative of a boundary. A well-known content-based approach is *TextTiling*, proposed by Hearst (Hearst, 1994). TextTiling is a content-based text segmentation algorithm that uses a sliding window approach to segment a text. The calculation is accomplished by two vectors containing the number of occurring terms of each block. The similarities between adjacent blocks within the text are computed to detect topic changes. The computed similarities are smoothed and used to identify topic boundaries by a cut-off function.

On the other hand, discourse-based techniques focus on story structure or discourse. These approaches make use of lexical features such as the presence of certain cue phrases that tend to appear near the segment boundaries. An example of discourse-based approaches is the Hidden Markov Model (HMM) segmentation method (Allan et al., 1998) that models "marker words", or words which predict a topic change.

2.3.3 Supervised and Unsupervised

A supervised text segmentation approach called *divSeg* was introduced by (Song et al., 2011), where they apply an iterative approach that splits text at its weakest point in terms of the lexical connectivity strength between two adjacent parts. After they found the weakest point in the text, their approach produces a deep and narrow binary tree. The tree is then flattened into a broad and shallow hierarchy through supervised learning of a document set or explicit input of how a text should be segmented. (Hsueh et al., 2006) described a supervised hierarchical topic segmentation approach that trains separate classifiers for topic and sub-topic segmentation.

On the other hand, (Eisenstein and Barzilay, 2008) proposed a Bayesian approach to *unsupervised* topic segmentation. They showed that lexical cohesion between text segments can be placed in a Bayesian context by modelling the words in each topic segment. (Malmasi et al., 2017) extended this model and segmented text based on the stylistically expressed characteristics of text such as change of authorship or native language. *TextTiling* (Hearst, 1994) and *C99* (Choi, 2000) are also considered unsupervised linear topic segmentation algorithms.

2.3.4 Borderline sentences detection methods

There are three main approaches to detect borderline sentences within text (i.e. sentences that identify the end or the beginning of a segment) (Labadié and Prince, 2008):

1- Similarity based methods: Represent text blocks as vectors and then measure the proximity by using (typically) the cosine of the angle between these vectors. The C99 algorithm (Choi, 2000) for example uses a similarity matrix to generate a local classification of sentences and isolate topical segments.

2- Graphical methods: Represent terms frequencies and use these representations to identify topical segments (which are dense dot clouds on the graphic). (Reynar, 1994) proposed a segmentation approach for locating text boundaries based on lexical cohesion

and a graphical technique called DotPlotting (Church, 1993). The approach is based on enumerating the lexical items in text and plotting points which correspond to word repetitions. Figure 2.1 depicts a sample dotplot of four articles. Since the repetition of lexical items occurs more frequently within regions of a text which are about the same topic or group of topics, the visually apparent squares along the main diagonal of the plot correspond to regions of the text.

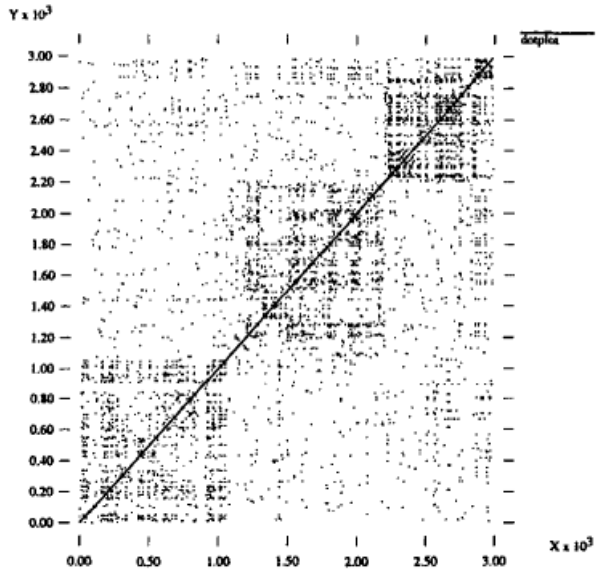


Figure 2.1 A dotplot of four concatenated *Wall Street Journal* (Reynar, 1994)

3- Lexical chains based methods: The central idea of approaches based on lexical chains is that if the text continues to use similar words, then it is probably still talking about the same topic (Manning, 1998). The notion of lexical chains was first proposed by (Morris & Hirst, 1991) to chain semantically related words together via a thesaurus. A chain links multiple occurrences of a term in the document and is considered broken when there are too many sentences between two occurrences of a term. (Wang et al., 2017) proposed a multi-granularity hierarchical topic-based segmentation system (MHTSS) that divides a *digital library* document into a document segmentation tree based on the structure of the document (built by its author) and the lexical cohesion between its paragraphs.

2.3.5 Linear and Hierarchical

If we look at the text segmentation from a text representation perspective, we can divide it into linear and hierarchical approaches. Linear text segmentation deals with the sequential analysis of topical changes where segments are non-overlapping and sequential. Linear text segmentation approaches focus on segmenting text into coherent segments where each

segment represents a specific topic (Badjatiya et al., 2018; Sakahara et al., 2014; Choi, 2000). It has been argued that this sequence model is sufficient for many purposes (Hearst 1994). An early linear text segmentation algorithm was the *TextTiling* approach introduced by Hearst (Hearst, 1994; Hearst, 1997). TextTiling applies linear text segmentation by measuring the lexical similarity between text blocks. Text blocks are the smallest units that constitute the text. They range from one sentence (Ye et al., 2008) to multiple sentences (paragraphs) (Kazantseva & Szpakowicz, 2014). TextTiling uses a sliding window to segment text. The calculation is accomplished using two vectors containing terms occurring in each block. The similarity between blocks is calculated by a cosine measure: given two text blocks b_1 and b_2, each with k token-sequences,

$$sim(b_1, b_2) = \frac{\sum_t w_{t,b_1} w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 \sum_{t=1}^{n} w_{t,b_2}^2}}$$

where t ranges over all the terms in the block, and w_{t,b_1} is the weight assigned to term t in block b_1.

(Galley et al., 2003) proposed *LcSeg*, a *TextTiling*-based algorithm that uses *tf-idf* term weights, which improved the text segmentation results. Another well-known linear text segmentation algorithm is *C99* introduced by (Choi, 2000). *C99* segments a text by combining a rank matrix, transformed from the sentence-similarity matrix, and divisive clustering. (Utiyama and Isahara, 2001) introduced a linear approach, *U00* that is based on language models, where they use Dynamic Programming (DP) and the probability distribution of words to rank and select the best segments. DP can be used to efficiently find paths of minimum cost in a graph. DP is used in text segmentation to represent each possible segment (e.g. every sentence boundary) as an edge providing a cost function that penalises common vocabulary across segment boundaries. (Misra et al., 2009) used Latent Dirichlet Allocation (LDA) topic model (Blei et al., 2003) to linearly segment a text into semantically coherent segments. Another approach that relies on LDA called *TopicTiling* was proposed by (Riedl and Biemann, 2012b). *TopicTiling* is based on the *TextTiling* algorithm, and segments documents using the LDA topic model. The algorithm represents segments as dense vectors of dominant topics based on terms they contain. (Eisenstein and Barzilay, 2008) proposed a Bayesian approach to unsupervised topic segmentation. They showed that lexical cohesion between text segments can be placed in a Bayesian context by modelling the words in each topic segment. (Naili et al., 2016) integrated a domain ontology in the topic segmentation in order to add external semantic

knowledge to the segmentation process. They proposed two topic segmenters called TSS-Ont and TSB-Ont based on C99 and TextTiling respectively. They used the same techniques as C99 and TextTiling but replaced lexical similarity with concept similarity. (Badjatiya et al., 2018) proposed a supervised neural network approach for text segmentation where they model the text segmentation problem as a binary classification problem. Given a document, they use the context of each sentence (i.e. sentences before and after it) for learning distinctive features for sentences that mark the beginning of the segment.

Most linear segmentation approaches can only produce single-level segmentation of a document. However, considering the structure of a document as a sequence of segments is in certain discord with most theories of textual content structure, where it is more usual to consider documents as trees (Grosz and Sidner, 1986; Mann and Thompson, 1988; Feng and Hirst, 2012; Kazantseva and Szpakowicz, 2014; Wang et al., 2017). Hence, hierarchical text segmentation is seen as a method that can represent a document as a tree-like hierarchy structure (Wang et al., 2017; Kazantseva and Szpakowicz, 2014; Eisenstein, 2009; Yaari, 1997).

Since one of the objectives of the research in this thesis is to understand unstructured text and build a representative structure out of it (RO 2), the research in this thesis focuses on the use of hierarchical text segmentation to perform this task. The following section reviews state of the art approaches for hierarchical text segmentation.

2.3.6 Hierarchical Text Segmentation Techniques

While linear text segmentation methods are concerned with splitting text into chunks of consecutive text fragments, hierarchical text segmentation methods attempt to iteratively split text into finer grained topic segments. Although it is widely believed that most documents display a hierarchical structure (Grosz and Sidner, 1986), work on hierarchical text segmentation is relatively sparse (Wang et al., 2017).

An early hierarchical text segmentation approach was proposed by (Yaari, 1997). Yaari used paragraphs as the elementary units for his algorithm and measured the cohesion between paragraphs using lexical similarity between them as the proximity test. An agglomerative clustering approach is then applied to induce a dendrogram tree over paragraphs where a segment corresponds to a subtree in the resulting dendrogram tree. The dendrogram is subsequently transformed into a hierarchical segmentation. Figure 2.2

shows a dendrogram of the *Stargazers* article that Yaari used as the test bench for evaluation.

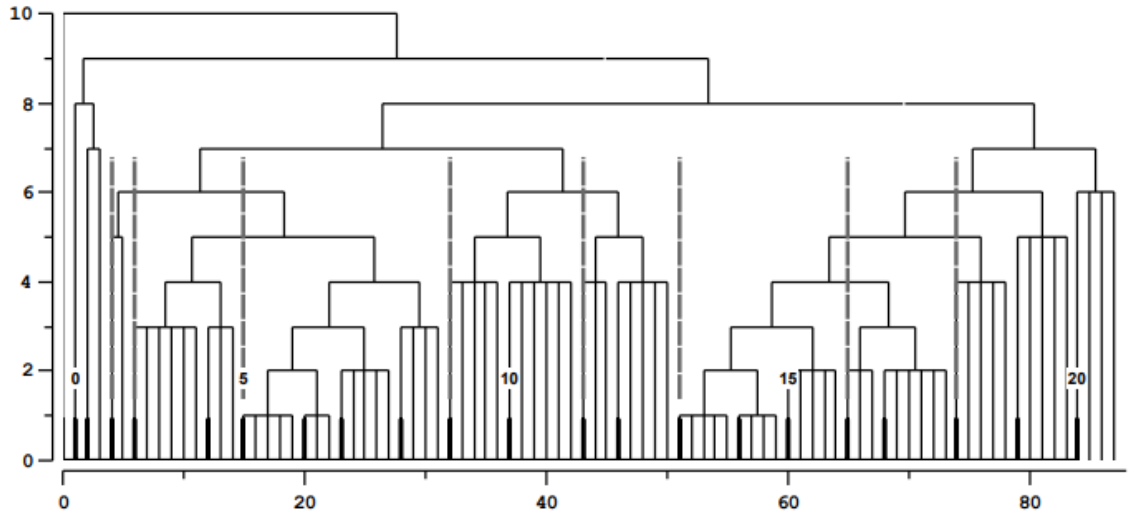


Figure 2.2 Paragraph dendrogram of the *Stargazers* article (Yaari, 1997)

The approach proposed by Yaari had been shown to be brittle as it requires a number of parameters that must be hand-tuned (Eisenstein, 2009). To overcome such limitation, (Eisenstein, 2009) proposed a novel unsupervised hierarchical text segmentation approach in which a Bayesian probabilistic framework that is based on LDA (Blei et al., 2003) was integrated. Eisenstein modelled each word token as a draw from a pyramid of latent topic models to create topical trees. The result of such work yielded an accurate and fast segmentation algorithm with a minimal set of tuneable parameters.

(Du et al., 2013) proposed an extension of the hierarchical Bayesian segmentation approach proposed by (Eisenstein, 2009) by considering more advanced topic models that model dependencies between (sub-) sections in a document. Although they utilised a hierarchical segmentation approach, they just used it to hierarchically model topics within document to improve the performance of linear segmentation, rather than develop hierarchical segmentation. (Slaney and Ponceleon, 2001) used an image segmentation algorithm (Leung et al., 2000) for hierarchical text segmentation. They extended the image segmentation approach by using Latent Semantic Indexing (LSI) (Landauer et al., 1998) to describe the position of a portion of the document in a multi-dimensional semantic space. (Angheluta et al., 2002) proposed a hierarchical text segmentation approach that applies a linear segmentation algorithm recursively to partition each major segment into a sequence of sub segments. However, they used the resulting segmentation in a summarisation system, and they evaluated the summarisation system but not the segmentation itself.

(Chien and Chueh, 2012) measured the topic similarity between sentences to form a beta distribution reflecting the prior knowledge of document boundaries in a text stream. The distribution of segmentation variables is adaptively updated to achieve flexible segmentation and is used to group coherent sentences into a topic-specific document. (Kazantseva and Szpakowicz, 2014) proposed HAPS, a hierarchical text segmentation approach that is based on a graphical model for hierarchical clustering called Hierarchical Affinity Propagation (Givoni et al., 2011). The input for HAPS is a matrix of similarity between text blocks (paragraphs in HAPS). HAPS requires the desired number of levels to be in the produced topical tree and a preference value for each data point and each level. HAPS also finds a centre for each segment at every level of the produced topical tree, a data point which best describes the segment. Recently, (Wang et al., 2017) proposed MHTSS, a multi-granularity hierarchical topic-based segmentation system which integrates features from lexical cohesion with document access structures (i.e. structure built by author) to build a composite framework. The system relies on the original structure of digital library resources (e.g. headings and subheadings) as the elementary units. Paragraphs within these sections are further segmented into subtopic segmentations based on lexical cohesion.

(Tsunoo et al., 2017) proposed a hierarchical text segmentation approach that captures the story structure of a broadcast news stream. The approach is a hierarchical model based on a word-level Recurrent Neural Network (RNN) sentence modelling layer and a sentence-level bidirectional Long Short-Term Memory (LSTM) topic modelling layer. Using the lexical tokens of each sentence, the approach starts to extract a vector embedding the sentence information in the word-level RNN layer. The output of this step is then used as the input for the bidirectional LSTM to model the sentence and topic transitions. After that, for each sentence, a topic posterior is estimated and a HMM follows to decode the story sequence and identify story boundaries.

2.3.7 Summary

This section reviewed a variety of approaches to text segmentation and highlighted the different categorisation criteria of this task. The section also reviewed the linear and hierarchical approaches to text segmentation.

However, regardless of the segmentation approach (linear or hierarchical), all the aforementioned approaches are limited by the fact that they can process only the information that they can ‘see’ (Cambria and White, 2014). In other words, they are based on the

lexical and/or syntactic representation of text, a method that relies mainly upon the traditional bag-of-words representation of text to measure similarity (or dissimilarity) between text blocks. However, a representation based solely on the endogenous knowledge in the documents themselves does not reveal much about the meaning of the text. Hence, the research in this thesis investigates the utilisation of external knowledge resources in order to enrich text and infer more information about text constituents.

2.4 Adaptive Systems

The previous sections reviewed a number of different NLP techniques, and focused on text segmentation as a technique for structuring text documents. NLP research has mainly focused on techniques that can be used to process text documents, but not, however, on how these techniques can be utilised to produce tailored content, adapted to the needs of individual users. Conversely, the field of content adaptation has primarily focused on methods and techniques of delivering adaptive content to individual users (Bunt et al., 2007). One area of content adaptation is Adaptive Systems. The aim of this section is to review adaptive systems, as content adaptation applications, to investigate how they process content to facilitate its discoverability and reusability.

2.4.1 Overview

There is an enormous increase in the amount of content available on the World Wide Web[2]. The architecture of the WWW has enabled the ease of content publication by millions of authors. However, the drawback to this ease of publication is that there is no organised method to catalogue or list the content contained in a collection of information nodes. Furthermore, the one-size-fits-all nature of web content makes it "same content for all people". With this static nature of content, users of an online news service, for instance, are provided with the same news regardless of their backgrounds or interests. As the number of users grows[3], their content needs become more diverse. This nature of content, and the increasing number of users, raised the need to change the way content is presented and delivered to individual users. Several research fields have emerged which focus on the challenges associated with the growing body of global content and the methods of delivering it to individual users. These challenges include: how to identify and

[2] There are over 51 billion web pages indexed by Google. Source: http://www.worldwidewebsize.com/ [Accessed October 2017].
[3] There are over 4 billion users on the internet. Source: http://www.internetlivestats.com/internet-users/ [Accessed September 2018]

retrieve content from different sources (Lawless, 2009); how to search for information in multiple languages (Ghorab, 2014); and how to deliver this content in a form that is most suitable for a specific user's or application's needs (Levacher et al., 2014).

As a result, the notion of Adaptive Hypermedia Systems (AHSs) emerged. In 1996, Brusilovsky introduced the first classification of Adaptive Hypermedia Systems (Brusilovsky, 1996). Since then, the field of adaptive hypermedia research has grown rapidly and many approaches and systems have been proposed (Brusilovsky and Pesin, 1998; Henze and Nejdl, 2001; Conlan and Wade, 2004; Brusilovsky and Henze, 2007; Staikopoulos et al., 2012; Aghoutane et al., 2017). These systems have tried to address the challenge of producing adaptive compositions from different information sources in order to deliver content in a form that is most suitable to an individual user. They have focused on providing such compositions based on a variety of user dimensions, such as user interests, prior knowledge, preferences or context. AHSs have successfully been used in a range of application areas from eLearning (Najar et al., 2016) to government portals (Penadés et al., 2014). The evaluations of AHSs have demonstrated their ability to allow users to achieve their goals faster (Steichen et al., 2011; Staikopoulos et al., 2012; Levacher et al., 2012c). This section therefore does not intend to provide an exhaustive list of adaptive systems or adaptation approaches, but instead to present the underlying structure of these systems and examine how they process content in order to tailor it according to the needs of individual users. A more detailed description of these systems, can be found in reviews carried out by (Brusilovsky, 1998) and (Knutov et al., 2009).

2.4.2 Anatomy of Adaptive Systems

Adaptive Systems have traditionally attempted to deliver dynamically adapted content to users through the sequencing of reconfigurable pieces of information. Many adaptive systems have been developed over the past 20 years. Since they were developed for different application areas (Brusilovsky, 1998), their architectural designs have diverged and hence, there is no consensus as to what the ideal architecture of such systems is (Knutov et al., 2009).

An attempt by Halasz and Schwartz has been carried out to capture the important abstractions found in a wide range of hypertext[4] systems (Halasz & Schwartz, 1990; Halasz &

[4] Although hypertext and hypermedia are often differentiated, this distinction is not made in their paper. They used the term hypertext generically to refer to both text-only and multimedia systems.

Schwartz, 1994). They proposed the Dexter model that provides a standard hypertext terminology coupled with a formal model of the important abstractions commonly found in a wide range of hypertext systems. Figure 2.3 shows the three layers and the two interfaces that the Dexter model consists of. The *within component* layer is concerned with the contents and structure within the "components", i.e. the links between content nodes in a single component. Components in the Dexter model correspond to one or more piece of content that can be in any form and from any source. A component could contain documents, chunks of text, graphics, images, animations, etc. The within component layer builds the components from the content source (hypermedia network for example) and builds the links between the contents within a single component. The interface that is responsible for addressing (referring to) locations or items within the content of an individual component is called the *anchor* interface.

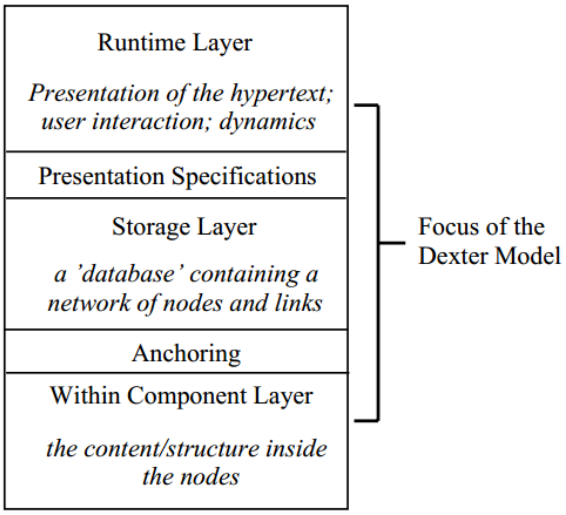


Figure 2.3 Layers of the Dexter model (Halasz & Schwartz 1990)

The produced components, from the *within components* layer, are then saved (indexed) in the *storage* layer. This layer describes a 'database' that is composed of a hierarchy of data containing components which are interconnected by relational links. The components are treated in the storage layer as generic containers of content.

After the content components have been built and indexed, the adaptive systems are then required to provide tools for the user to access, view, and manipulate the network structure. This functionality is captured by the *runtime* layer of the model. This layer is concerned with how the content would be presented to the user of the adaptive system. The *presentation specifications* interface is the mechanism that connects the storage layer with the runtime layer. This interface contains the information about how a component is to

be presented to the end user. Figure 2.4 depicts the three layers of the Dexter model as embedded in an actual adaptive system.

Although the Dexter model provides only a bare-bones model of the mechanism for presenting content to the user for viewing and editing, it represents a good reference for the most AHSs which came after it because it encompasses most of the components currently encountered in modern AHSs.

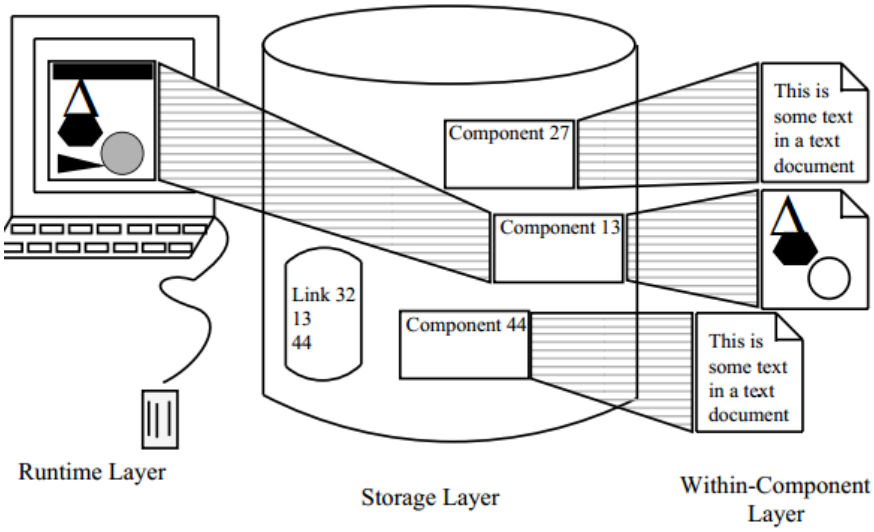


Figure 2.4 A depiction of the three layers of the Dexter model as embedded in an actual adaptive system.

Another generic architecture for AHS called the AHAM model was proposed by De Bra in 1999 (De Bra et al., 1999). AHAM (Adaptive Hypermedia Application Model) is based on the Dexter model. As depicted in Figure 2.5, AHAM augments the *storage* layer in Dexter by adding three (sub) models: *user model*, *domain model* and *teaching model*. Each of these models contributes to the content adaption process. The user model keeps track of evolving aspects of the user, such as preferences and domain knowledge. 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 (Dimitrova, 2003). The domain model describes how the content is structured and linked together while the teaching model consists of pedagogical rules. These rules define how the other two models are combined to provide ways to perform the actual adaptation.

Some approaches have tried to extend the AHAM model or provide a new one. For instance, the Munich model (Koch and Wirsing, 2002) used the Unified Modelling Language (UML) to capture all major parts of the adaptive system architecture.

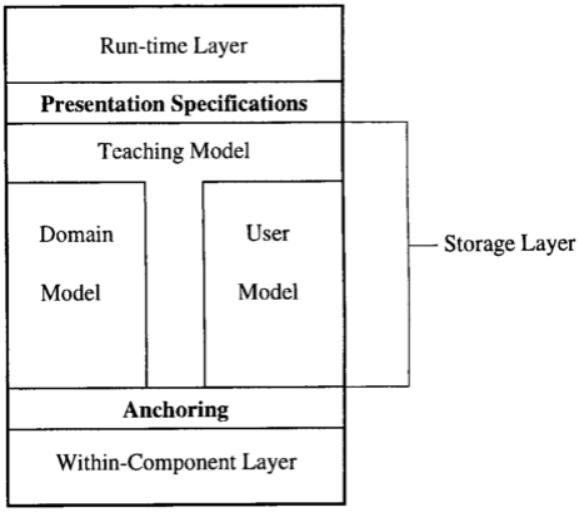


Figure 2.5 AHAM model (De Bra et al., 1999)

Over time, the structure of AHSs has evolved and the main focus of AHSs was on how to apply and improve separation of concerns between preeminent components of this structure (Brusilovsky, 2001). APeLS (Adaptive Personalized eLearning Service) (Conlan et al., 2002) for example, was developed as a service to deliver personalised educational courses based on a multi-model, metadata driven approach. It proposed an additional separation of concern regarding the pedagogical aspects of AHSs encapsulated within a narrative model. A core module in APeLS is the adaptive engine that consolidates selected narratives and content to facilitate the reuse of learning resources across different pedagogical models.

As it can be seen, with the evolution of the AHSs and their models, separation of concerns between preeminent components of adaptive systems has also evolved. More recent approaches has brought this separation of concern yet a step further (Steichen et al., 2011; Keeffe et al., 2012; Staikopoulos et al., 2012; Bayomi, 2015). (Levacher et al., 2012b) for instance, shows how even concerns related to content models can be delegated to an external web service.

2.4.3 Models of Adaptive Systems

As the second research question posed in this thesis focuses on the discoverability and reusability of content for adaptive systems (section 1.2), it is important to understand how content is handled by adaptive systems. Therefore, this section briefly reviews different models that adaptive systems contain and focuses on the content model which describes how content is organised within adaptive systems.

With their evolution over time, adaptive systems have incorporated and modified different (sub) models in order to provide customised content that is tailored to their users' interests, knowledge and goals (Kardan et al., 2015). As illustrated in the previous section, different reference models have been proposed to provide a generic architecture for adaptive systems. Although there is variation between models, there is a set of sub-models that they all have in common and are essential for any adaptive system (Knutov et al., 2009; Wilson and Scott, 2017). These sub-models can be classified, based upon what they are concerned with, into:

Domain Model: describes how domain knowledge is organised within an adaptive system by defining all concepts relevant to the domain, as well as the relationships between these concepts. It is a description of the application which contains facts about the domain, i.e. the objects, their attributes and the relationships between objects (Benyon & Murray, 1993). Domain models usually represent knowledge in the form of a structured or hierarchical set of topics (Fiqri & Nurjanah, 2017), in the form of a network or a graph, that comprehensively organises concepts and all the relationships between them (Sosnovsky and Brusilovsky, 2005; Gandara et al., 2014).

User Model*:* is an explicitly represented collection of data about the user which allows the system to tailor its content to the user's needs (Thaker et al., 2018; Vassileva, 1996). It is used to represent information about, and the characteristics of, the user. These characteristics are updated through the interaction of the user with the system (Brusilovsky 1998). Its main role is to keep track of evolving aspects of the user, such as preferences, goals, knowledge, learning style and other relevant aspects (Brusilovsky and Millán, 2007). (Finin, 1989) stated that a user model is *'knowledge about the user, either explicitly or implicitly encoded, which is used by the system to improve the interaction*.' Knowledge within a user model usually refers to keywords (Ahn et al., 2007) or concepts (Dimitrova, 2003; Dimitrova & Brna, 2016) defined by the domain model and can be updated by rules specified within the adaptive engine. These rules guide the user towards interesting new information and keep the user away from information that is considered not to be appropriate or relevant.

Adaptation Model: Content adaptation is the main task of any adaptive system. At the heart of the adaptive system is the *adaptive engine* that is responsible for performing the content adaptation according to the adaptation rules specified in the adaptation model (Wu et al., 2000). These rules specify how the user's knowledge (from the user model)

influences the content presentation from the domain model. (Wu et al., 2000) divided the adaptation process in adaptive systems into two levels: 1) author level where an author (a domain expert) writes the adaptation rules and 2) system level where the system designers build an adaptation engine to apply these rules.

Content Model: describes how content is organised in the adaptive system. It represents what resources are available to the system and how they are connected to each other. Resource descriptions within this model enables the delivery of content based upon its match with various combinations of requirements, provided by the adaptive system. A content repository is used as a basis for the content model where content resources are indexed to be addressed, later, through metadata annotations that describe the type and properties of resources available within this content repository (Maycock and Keating, 2017).

This section highlighted the four models that are considered the main skeleton of any adaptive system. However, other external models that exist can be integrated in the adaptive system to perform a specific task. Examples of such models are: knowledge models, pedagogical models, usage models, etc. (Brusilovsky & Henze, 2007).

2.4.4 Content Models

As mentioned above, the content model describes how content is organised in order to enable the delivery of resources which match various combinations of content requirements provided by an adaptive system (Levacher, 2014). These content requirements result from the personalisation experience intended to be produced for a given individual user.

In order to supply a set of content resources in a form which is deliverable to an adaptive system, this content is required to be available within a given content model. The role of this model is to enable the delivery of content resources (available within a content repository) which match the content requirements of an adaptive system. Furthermore, content resources within the content repository should be augmented with additional information to support its discoverability. This additional information is referred to by (Brusilovsky & Henze, 2007) as *adaptation-specific information* and is classified into four subcategories:

1- *Attribute information:* consists of metadata that describes the type and properties of a content resource. Such metadata information is used to identify and retrieve the content resources available within the content repository that are matching as close as possible the content requirements requested by the adaptive system. (Henze & Nejdl, 2001) added a metadata layer to describe content in KBS-Hyperbook. The content in KBS-Hyperbook is marked as "introductory", "quiz", "example", etc. The ARCHING system (Adaptive Retrieval and Composition of Heterogenous INformation sources for personalised hypertext Generation) (Steichen et al., 2011) also describes resources based upon properties such as "language", "size" etc. (Maycock and Keating, 2017) used a *Content Analyser* module in their adaptive system to automatically generate metadata for content resources to describe the cognitive impact that the resource would have on a learner.

2- *Inter-document information:* This type of information assigns relationships between content resources in order to construct a hyperspace network of these resources. This type of information is used in order to facilitate user navigation between the different content resources available within the content repository (Steichen & Wade, 2010).

3- *External model connection:* Different external models can be added in order to augment the knowledge about the content in the content hyperspace. Examples of these external models are conceptual, pedagogical or goal models (Brusilovsky & Henze, 2007). Connection to these external models supports different content presentation techniques such as concept-based sequencing techniques (Staikopoulos et al., 2014). Content resources in the InterBook adaptive system (Brusilovsky et al., 1998), for example, are connected to a domain model with links that identify whether the domain concepts assigned to them are outcomes or prerequisites to this resource.

4- *Intra-document information*: This provides information about the internal characteristics of a content resource such as a topic that is covered by a content resource (a document or a fragment). Such information allows the adaptive system to select a content resource that is most relevant to the user's goal, knowledge or preferences (Weal et al., 2007). This technique helps individual users focus their attention upon the most relevant information presented to them.

Content models can be classified based on the type of content they provide. Generally, they can be classified into two main types: Closed corpus and Open corpus content models (Levacher, 2014).

2.4.4.1 Closed Corpus Content Models

The main objective of the early generation of adaptive systems is to overcome the problems associated with the inflexible and static nature of content, and to find techniques that could be used to tailor this content according to users' preferences (Chesnais et al., 1995; Brusilovsky et al., 1996; De Bra & Calvi, 1998; da Silva et al., 1998). This objective meant that these systems primarily focused on content tailoring techniques, rather than the production of the content itself. As a result, the content models of these systems were based on content resources that were manually handcrafted in order to support the adaptation techniques in each individual adaptive system (De Bra and Calvi, 1997; Maycock and Keating, 2017). Such content models are referred to (in this thesis) as *closed corpus* content models.

In a closed corpus content model, resources, their attributes and relations to other resources are *known* at the design time of the adaptive system[5] (Brusilovsky & Henze, 2007). Resources within this model are predictable and static in the sense that resources and relationships between them possess pre-determined content as well as a common structure, known at design time by systems consuming these resources. Figure 2.6 shows the architecture of an adaptive system that operates on a closed corpus content model where the content is available within the system (Aroyo et al., 2004). The Content model (content layer) captures the content as it is stored in the content repository along with its description then makes it available to the Application layer that provides the adaptive content to the system's users.

Systems that operate on a closed corpus of resources are only able to work within a limited set of documents that have been manually structured and indexed (Dieberger & Guzdial, 2003; De Bra et al., 2003). Although closed corpus content models are limited in the amount of resources they can provide, the content they do have is well formed and curated. This is because the content resources within these models are usually collected and curated manually by domain experts. This in turn enables the adaptive system to be supplied with the ideal resources that are needed to deliver adaptive content.

[5] Content resources are not necessary to be known at the design time of the adaptive system, it can be known at the design time of a course or experience that the adaptive system can run.

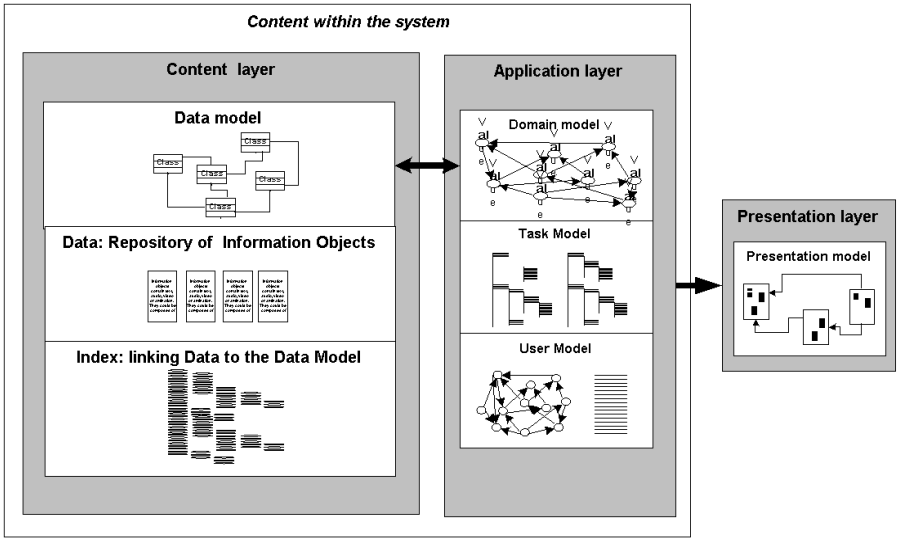


Figure 2.6 An Adaptive System with a closed corpus content model (Aroyo et al., 2004)

However, although the benefits of closed corpus models have been demonstrated within different adaptive systems (De Bra & Calvi, 1997; Dieberger & Guzdial, 2003; De Bra et al., 2003), they have failed to overcome the problems associated with extending and updating the content resources (Levacher et al., 2011). This is because the inherent nature of content resources in these models are built manually. As a result, they do not scale and are impractical for most real-world applications. Extending the resources in such models is a labour-intensive task that needs much time and effort from a domain expert. Additionally, whenever a change in the structure or presentation of content resources needs to be made, adaptation techniques and algorithms developed within the adaptive system must be altered accordingly or replaced altogether, which can be quite labour intensive. Such high time, effort and cost requirements associated with closed corpus models result in a limited volume and diversity of resources, available for delivery to adaptive system (Conlan et al., 2002).

Another limitation of closed corpus content models is that as they are mainly developed for a specific system, according to its needs, their closed nature prevents their content from being reused within different adaptive systems. A content resource that is created and indexed in a content repository for a specific adaptive system is only applicable for use within that system. It is indexed based on the characteristics, requirements and adaptation techniques developed for that system. Hence, reusing such a resource within another system requires an inherent modification to that resource which is considered a time-consuming and effort-intensive task.

Over time, and with adaptive systems serving a wider range of users with more personal needs, the goals and preferences of the system's users change. As a result, the content resources supplied to the adaptive system need to be continuously maintained in order to cope with these new information needs. Since closed content models are only able to function over sets of content resources processed and prepared at design time, maintaining such resources is a labour intensive task (Brusilovsky & Henze, 2007). Furthermore, this creation of content resources a-*priori* of system deployment entails the need to predict the type and quantity of resources, which will be needed by the adaptive system (Steichen et al., 2011).

2.4.4.2 Open Corpus Content Models

Various adaptive systems have tried to overcome the limitations encountered with the use of closed corpus content models (Zhou et al., 2007; Henze & Nejdl, 2001; Conlan et al., 2013). These systems utilise content models that can easily incorporate content from different sources in order to allow the adaptive system to serve a wider range of content to different users. Such content models are referred to (within this thesis) as open corpus content models.

In open corpus content models, resources, their attributes and relations to other resources are *unknown* at the design time of the adaptive system and, moreover, can constantly change and expand (Lawless, 2009; Steichen, 2012; Staikopoulos et al., 2014). As a wealth of diverse information has now become accessible on the WWW, it becomes the largest repository for open corpus content (Weal et al., 2007). The inherent nature of the Web expanded the range of content adaptivity offered by adaptive systems as this content comes in different languages, covers an unrestricted set of domains and is available in different formats (Levacher et al., 2009). Furthermore, by exploiting the abundant content resources available on the web, the content provided by the adaptive system can always be kept up-to-date.

Recently, adaptive systems have moved towards exploiting content available on the Web (Steichen et al., 2009; Steichen & Wade, 2010; Levacher et al., 2012c; Levacher et al., 2014). These systems exploit Web technologies to retrieve content from the sources reachable through the Web and deliver it to their users. This increased the volume of content available to such systems and hence allowed them to serve a wider range of users. This also significantly reduced the amount of manual labour involved in developing content.

The process of developing content within the closed corpus content supply models (section 2.4.4.1) involves the addition of adaptation-specific information to content resources, e.g. KBS-Hyperbook (Henze & Nejdl, 2001) and the adaptive system proposed by (Maycock and Keating, 2017). However, open corpus content resources do not provide any adaptation-specific information since they are harvested on-the-fly and are unknown at design time (Steichen et al., 2011). Hence, adding such information can only be performed at run-time (Knutov et al., 2009).

As a result, early research on harvesting open corpus content has focused on the techniques and methods which support the addition of adaptation information to content resources after the adaptive system have been deployed. This task is very challenging as adding such information would require the system designer to anticipate all the potential requests that would be submitted by the adaptive system. Such requests would be influenced by the status of users who use the adaptive system. These influences, on a request, are derived from the changes and the fluctuations in the users' preferences and goals over time.

Since the early research has primarily focused upon the elaboration of techniques which support the addition of such information to content resources, techniques that have been developed have tended to use content resources in the form they were created in. Such techniques can be divided into three main categories:

1- *User-Supported Incorporation Techniques*: In these techniques, adding new content resources from the open corpus can be achieved directly by individual system users (Carmona et al., 2002; Henze & Nejdl, 2001). Adaptive systems that use these techniques rely on a pre-existing set (a closed corpus) of content resources. These resources are then augmented by providing the individual users of the system with the tools needed to annotate and link external resources from the web (Smits & De Bra, 2011). Examples of such systems are: KBS Hyperbook (Fröhlich et al., 1998; Henze & Nejdl, 2001), SIGUE (Carmona et al., 2002) and Knowledge Sea II (Brusilovsky et al., 2004).

2- *Keyword Incorporation Techniques*: Since user-supported techniques are limited in that they require a large amount of manual effort, some adaptive systems started to incorporate different techniques that can automatically incorporate content resources from the Web (Ahn et al., 2007; Zhou et al., 2007; Zhou et al., 2008). These systems exploit the capabilities of classical Information Retrieval (IR) approaches in order to

find appropriate content resources available on the Web. Such systems rely on keyword similarity algorithms to create links between content resources (Zhou et al., 2008). An example of such systems is the ML-Tutor system (Smith & Blandford 2003).

3- *Semantic Metadata Incorporation Techniques*: These approaches relied upon extracting metadata from open corpus content resources (such as difficulty, narrative cohesion, interactivity type or interactivity level) in order to assign such metadata to each resource incorporated within the adaptive system (Şah & Wade, 2010; Hargood et al., 2011; Şah & Wade, 2012). In this approach, the metadata representation of a document is a set of concepts (in contrast to keywords) that are part of a domain ontology (Millard et al., 2003). These concepts represent the topics covered in each individual document. This semantic representation of content means that this approach provides improved structuring of the resulting hyperspace, when compared to keyword-based techniques (Brusilovsky & Henze, 2007). This semantic representation of content resources has also extended to the content indexing process (Brusilovsky et al., 1998; Sosnovsky et al., 2012). This in turn enabled open corpus documents to be organised into hierarchies, which further improves the concept-based sequencing navigation along the structure of these ontologies. (Ye et al., 2010) presented an approach that incorporates open corpus resources from the CiteSeerX[6] website where they classify each resource based on a predefined ontology.

2.4.5 Summary

This section presented an overview of adaptive systems, as an application for content adaptation, and reviewed their anatomy, their models and in particular their content model. Closed and open corpus content models were reviewed in order to better illustrate how adaptive systems process the different types of content.

The third objective of this theses (RO 3) is to build a content-supply service to enhance content discoverability and reusability for adaptive systems. In order for this service to overcome the limitations of the state of the art content-supply approaches, there is a need to understand how these approaches employ different techniques to discover and reuse content. The following two sections therefore focus on reviewing content discoverability and content reusability techniques.

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

2.5 Content Discoverability Techniques

One of the main services that adaptive systems offer to their users is the provision of content that is tailored to individual user's needs. In order to provide this service, adaptive systems utilise different techniques to incorporate content that meets the requirements of their users (section 2.4.3). This requires these adaptive systems to have the ability to easily discover content that matches the goals and requirements of their users.

Section 2.4.2 of this chapter demonstrated how the anatomy of adaptive systems has evolved over time. Through this evolution, different models were added to enhance the content adaptability in these systems. From the different architectures that have been proposed to capture the important abstractions found in adaptive systems, it can be seen that content discoverability, in adaptive systems, typically relies upon two phases. The first phase is called (in this thesis) *external content discoverability*. This phase involves finding content resources that are deemed to be relevant to the domain of the adaptive system. The discovered content in this phase is collected (typically) in its native form. This content is then indexed in a content repository (Storage Layer in Figure 2.3, Figure 2.4 and Figure 2.5) inside (or outside) the adaptive system to be used later according to the adaptation techniques of the system.

The second phase thereafter matches individual requests submitted by the adaptive system's users with the most relevant content resource previously discovered and annotated in the first phase. This second phase is called (in this thesis) *internal content discoverability*.

This section highlights the different techniques used by adaptive systems in order to externally (section 2.5.1) and internally (section 2.5.3) discover content resources that are relevant to their users' needs. The section also discusses the different indexing techniques (section 2.5.2) used by adaptive systems in order to index their content in a manner that facilitates its discoverability according to the user needs.

2.5.1 External Content Discoverability Techniques

As content incorporation is vital for adaptive systems, the process of discovering content which is relevant to the needs of the adaptive system's users is critical. As discussed in section 2.4.4, there are two types of content models in adaptive systems: closed corpus and open corpus content models.

For adaptive systems that operate on a closed corpus of content (section 2.4.4.1), the resources, their attributes and the relations to other resources are *known* at design time. Hence, they do not need to utilise approaches for discovering content (De Bra and Calvi, 1998; Maycock and Keating, 2017). Content discovery (and incorporation) in these systems is completed, in the majority of cases, by the content author or a domain expert at design time. Despite the fact that incorporating content in these systems is straightforward and does not require effort to externally discover (as it is prepared by content author), the manual work involved in the incorporation process is quite labour intensive. Furthermore, these systems are only able to work with a limited set of documents that have been manually structured and indexed (Dieberger & Guzdial, 2003; De Bra et al., 2003).

In adaptive systems that operate on an open corpus of content (section 2.4.4.2), the resources, their attributes and the relations to other resources are *unknown* at design time and can constantly change and expand (Lawless, 2009; Steichen, 2012; Staikopoulos et al., 2014). As a result, these systems must utilise external content discovery approaches in order to cope with the inherent diversity of open corpus content.

Incorporating a wide range of open corpus resources is pointless if these resources are of no relevance to the needs of the adaptive system. As the discoverability of web resources is a vast field of its own and extensively studied[7] (Manning et al., 2008; Steichen et al., 2012; Ghorab et al., 2013; Onal et al., 2018), this section aims to focus on the techniques used by open corpus adaptive systems in order to find content which is relevant to their needs and domain. Additionally, since the focus of this thesis is on how adaptive system process content and index it in a manner that makes it amenable for discovery and reuse, this section intends to present the reader with a brief overview of the fundamental approaches for external content discoverability.

External content discoverability techniques can be classified into two main approaches:

1- *Standard Content Discoverability Techniques*: Adaptive systems have mostly relied upon either the manual incorporation of open corpus resources (Henze and Nejdl, 2002), or the use of standard IR mechanisms to do so (Aroyo et al., 2004). While manual incorporation of open corpus content is usually carried out by individual users of the system, IR techniques are considered the most efficient method of largescale content discovery. The ArtEquAKT system (Millard et al.,

[7] In general, the field that is concerned with techniques and methods of finding (discovering) resources on the open web according to a user query is called Information Retrieval (IR).

2003; Weal et al., 2007) utilises a traditional search engine to discover open content resources that comprise biographical information about artists. The Personal Reader (Dolog et al., 2004) uses Lixto (Baumgartner et al., 2001) for standard web crawling and Edutella (Nejdl et al., 2002) a peer to peer search system, to discover and incorporate open corpus resources. (Meng et al., 2017) use results obtained from a search engine (Google search) to retrieve websites that present detailed content regarding a specific course.

2- *Focused Discoverability Techniques (Focused Crawlers)*: Although the integration of standard IR techniques within adaptive systems is relatively easy, these techniques are considered general purpose discoverability mechanisms and do not specialise in specific areas of interest. As a result, some adaptive systems started to utilise focused crawlers to extend these techniques by enabling the discovery of content, which meets pre-determined classifications (Steichen et al., 2009; Levacher et al., 2012a). The goal of the focused crawler (also referred to as topical crawler) is to selectively seek out pages that are relevant to a predefined set of topics (Lawless, 2009). (Steichen et al., 2009) proposed an educational adaptive system which is built on top of the APeLS system (Conlan et al., 2002) and uses the autonomous Open Corpus Content Service (OCCS) focused crawler (Lawless, 2009) to incorporate open educational material available on the WWW. Another example of a system that use focused crawler is Slicepedia (Levacher, 2014) that uses the 80Legs[8] web crawler as the harvester module.

2.5.2 Content Indexing

The indexing mechanism is a critical component of any IR-based system, which provides a formalised, simplified and machine usable representation of content contained within each resource (Salton, 1989). In adaptive systems, the indexing mechanism varies depending upon the nature of the content resources being incorporated in each individual system[9]. In adaptive systems that rely on a closed corpus of content, the content is structured and annotated by the content author (or a domain expert), therefore indexing this content is relatively straightforward, as it does not require an automatic approach for this task (Aroyo et al., 2004). However, manual indexing of content resources is a labour-

[8] http://80legs.coms
[9] In e-learning domain, indexes for learning resources are usually called learning object repositories (LOR)

intensive task. On the other hand, adaptive systems that rely on open corpus content require an automatic approach to index the harvested content.

Content indexing, in adaptive systems, can be classified into two main approaches: *document-level* (De Bra & Calvi, 1998) and *fragment-level* (Weal et al., 2007; Levacher 2014).

2.5.2.1 Document-level Indexing

In *document-level* approaches, the harvested resources are indexed in their native form (usually as HTML web pages) as one-size-fits-all document level granularity resources. For example, within the KBS Hyperbook (Fröhlich et al., 1998; Henze and Nejdl, 2001), once identified by system users, open corpus resources are indexed in a content repository. Each resource is assigned to a knowledge concept of the application domain such as the "*if*" or "*while*" concepts in a programming language ontology. These assigned concepts are used for indexing the content resources in the *storage module*. After that, links between existing (closed corpus) resources and the newly indexed (open corpus) resources are generated automatically based upon the concept that each new resource was assigned. The indexed documents are then adapted and presented based on concepts that represent the user's goals. Another example is the ML-Tutor system (Smith and Blandford, 2003). ML-Tutor utilises an IR mechanism to collect content resources from the open web. The harvested documents are then indexed along with the prominent keywords in each document. A document is then presented, in its native form, to the user based on the similarity between the document's keyword vector and the keyword vector in the user's model.

2.5.2.2 Fragment-level Indexing

On the other hand, *fragment-level* approaches focus primarily on content where the adaptation is performed at a finer level of granularity (Bunt et al., 2007). In these approaches, the harvested resources are processed and segmented into coherent fragments. These fragments are then indexed in the content repository. ArtEquAKT (Millard et al., 2003; Weal et al., 2007) for example, utilises information extraction and knowledge management techniques to create dynamic biographies of artists from content available on the web. The system relies on a traditional search engine to harvest content resources that comprise biographical information about artists. The harvested resources are then fragmented into paragraphs (and sentences) that are analysed syntactically to identify whether

it contains any relevant information about the artist requested by individual users. These fragments are then associated with the relevant instances in a Knowledge Base (KB) and indexed in a MYSQL database to be used later in generating biography pages. Figure 2.7 shows how ArtEquAKT extracts the different fragments from webpages to create dynamic biographies.

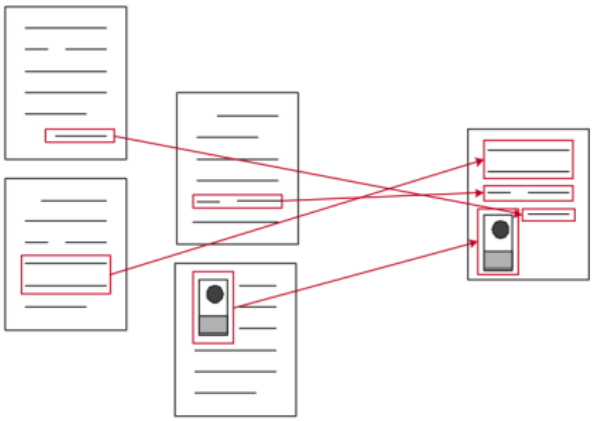


Figure 2.7 The ArtEquAKT system

Another example is PMCC (Steichen, 2012), that delivers a personalised content to individual users using open corpus content as fragments of text. The system utilises an automatic metadata extraction technique to enrich the harvested content. This enriched content is then fragmented using a wrapper-based content fragmentation approach (Bunt, Carenini and Conati, 2007) to identify regions of pages in order to produce individual fragments of content. These fragments are then presented to the users based on their knowledge in user models.

Slicepedia (Levacher et al., 2014) applies the Densitometric Content Fragmentation approach (Kohlschütter & Nejdl, 2008) to fragment the harvested open corpus content into segments based on their HTML structure. The fragments (called slices) are then indexed in a content repository along with concepts that represent the topics covered in each fragment. The indexed fragments can then be retrieved based on a request from an arbitrary adaptive system. The architecture of Slicepedia is depicted in Figure 2.8.

Both approaches (document-level and fragment-level) are limited in their ability to provide different levels of granularity for the indexed content. Document-level indexing approaches limit the extent to which these resources can be modified or recomposed together. This in turn hinders the indexed content from being reused in multiple systems. Furthermore, as pointed by (Bunt et al., 2007), presenting the incorporated open corpus resources in their native form allows more content to be "*visible to the user. However,*

the more content is shown, the higher the chance of generating information overload and reducing attention to the most relevant information, defeating one of the very reasons for having adaptive systems in the first place".

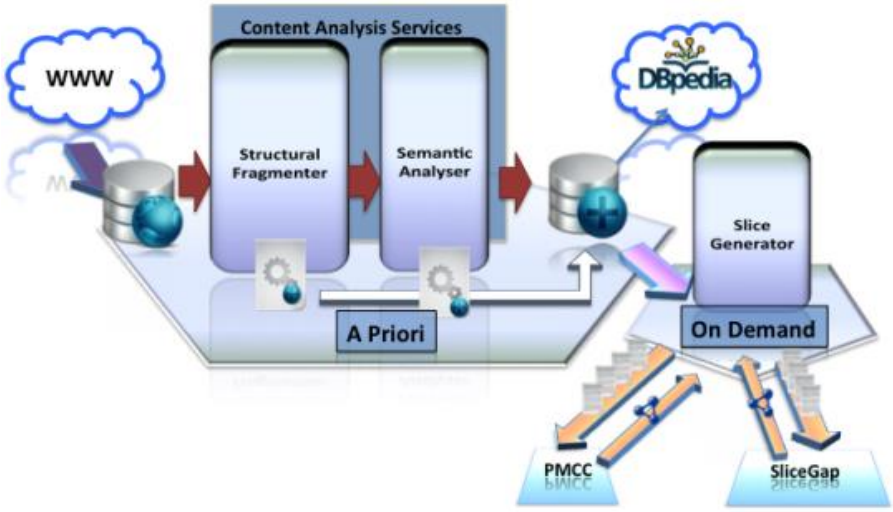


Figure 2.8 Slicepedia Architecture Pipeline

For fragment-level indexing approaches, relying upon the original structure of the content resource (HTML structure or paragraphs structure) to produce fragments and hence index them, does not reflect the needs or preferences of individual users or applications. This is because the structure posed by each resource reflects the needs and the point of view of its author. While each adaptive system has its own content requirements (based on its users), relying upon such structure does not reflect these requirements. Furthermore, in these approaches, the indexing process means that the final structure of each content item is already built. This limits the capability of these approaches to change the structure according to the individual user or application needs.

Another limitation of both approaches is the conceptual coverage associated with each indexed content item (weather a full document or a fragment). In other words, these approaches annotate individual content items with a small number of concepts (or sometimes one concept) that represent the topic(s) covered by that content item. However, a content item could be associated with many different concepts with different relevancy levels that represent to what extent each concept is covered by that content item. By only considering a few concepts that are deemed most relevant and ignoring the less relevant concepts, these approaches ignore a range of possible interpretations of that content item. This in turn hinders this content item from being properly discovered and presented to users according to their needs, and from being reused in other systems.

Hence, there is a need to index content in its smallest granularity, without only considering the original structure that has been built by the author of the content resource. Ideally, content should be indexed at a range of granularity levels and associated with all relevant concepts at each of these levels. The research in this thesis hence tries to overcome the limitations of these aforementioned approaches and tries to build a structural hierarchy out of text documents without relying on their original structure. After building such a structure, the documents are indexed in a manner that facilitates ease of discovery and reuse.

2.5.3 Internal Content Discoverability Techniques

Adaptive systems have traditionally attempted to deliver dynamically adapted content to their users through the sequencing of reconfigurable content items. As mentioned previously, these content items can be a full document (the whole content resource), or a fragment extracted from a content resource. Once the most relevant content items are harvested and indexed, they need to be effectively discovered from within the repository of resources maintained by the adaptive system, or the repository of resources maintained by content-supply service that provides content to the adaptive system (Levacher et al., 2014). Discovery of this indexed content involves deciding what content is most relevant to the needs or goals of current user (Bunt et al., 2007). Generally, strategies for content discovery compute a measure of relevance for each content item (e.g. fragment) to the target user's model (Steichen et al., 2011; Sosnovsky et al., 2012; Şah & Hall, 2013).

The ArtEquAKT system (Millard et al., 2003; Weal et al., 2007) provides human authored story (biography) templates that are written in the Fundamental Open Hypermedia Model (FOHM) (Millard et al., 2000). Templates are structured in XML format and saved in the contextual structure server, Auld Linky (Michaelides et al., 2002). Each leaf of the structure is a query which resolves into either a statement from the extracted information (stored in the Knowledge Base) or a reference to an original text fragment (stored in the database). Each query uses the vocabulary of the ArtEquAKT ontology to discover fragments of text concerned with a particular aspect of the biography.

KBS Hyperbook (Fröhlich et al., 1998; Henze & Nejdl, 2001) presented an ontology-based user and content modelling approach. This system structures a domain into a set of concepts and their relationships. A concept is assigned to each content item (page) in the system from that structure and the user model is constructed from the same structure based on the knowledge that the user has. By classifying the pages into these concepts,

navigation links are inferred automatically from the relationships amongst concepts. Based on a concept in the user's learning goals (according to the user's knowledge), the adaptation module in the system queries the storage module (content repository) for content items that match this concept. If a text unit should be presented to the user, dynamically generated relations to examples and other information, e.g. to the Sun Java tutorial, are returned.

(Ghauth and Abdullah, 2011) used collaborative filtering and content-based approaches in their proposed e-learning recommender system. Before the recommendation process begins, the learning materials are uploaded by the instructors along with keywords that describe each item. Once the material is indexed, the recommender system uses the manually entered keywords to query the content repository for other learning materials. The keywords attached to each item are then used to calculate the items' similarity.

(Farrell et al., 2004) proposed an eLearning system that automatically generates individualised learning paths from a repository of web resources. They proposed the notion of Dynamic Assembly that is based on connecting relevant search results and sequencing the selected learning objects on a learning path. The process is based upon the learner's keyword query, desired level of detail, and optional desired course duration. The system utilises a search engine that uses the set of keywords entered by the user to discover the relevant resources that have been previously indexed in a content repository. The system also provides users with the capability of identifying the Search Scope. A Search Scope of "*overview*" explores related topics, while a Search Scope of "*indepth*" focuses primarily on a single topic. Advanced query options allow users to restrict the search for learning objects to particular resource types, levels of difficulty, and other preferences.

In Slicepedia (Levacher et al., 2014), after analysing the harvested open corpus resources and indexing them as fragments in the content repository, the system employs a *slice searcher* module that allows slice consumers (content consumption applications) to specify a list of keywords and/or DBpedia concepts to be sent as part of a SliceQuery object. This object is used to query the content repository in order to discover fragments that match the different criteria specified in that object. Fragments discovered through both a conceptual and keyword search are then subsequently merged into slices, depending upon the granularity requirements in the request.

All these approaches, however, are limited in that they rely on the bag-of-words representation of content in measuring similarity between a content item and the user (or application) query. Even systems that use concepts in this task (e.g. Slicepedia), mainly rely upon a limited conceptual representation of content items (i.e. using one or very few concepts). Furthermore, as mentioned earlier (section 2.5.2), such limited conceptual representations in turn hinder content items from being properly discovered and presented to the users according to their needs, and from being reused in other systems.

2.6 Content Reusability Techniques

Section 2.5 presented different techniques used by adaptive systems in order to support the external and internal discoverability of content. The section also discussed the different techniques used for indexing that content. This section, on the other hand, explores more general aspects of content reusability. In particular, it examines the various forms of content reuse which have been used to maximise the value of existing resources.

The production and delivery of content has traditionally been a very linear process (Lawless, 2009) that requires the painstaking authoring of content by a domain expert for a specific purpose and needs of an individual application (Meyer et al., 2011). This in turn limited the use of such content within an individual application and hindered its reusability within other applications. Thus, the repurposing and reuse of content resources became, and remain, major challenges. To reduce the content production overhead on the content author it is imperative to facilitate the maximum reuse of content resources (Dagger et al., 2002). As stated by (Levacher et al., 2014) "*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.*" As a result, the notion of content reusability has emerged which tried to go beyond traditional content production and delivery methods by improving its discoverability and reusability between potential content consumers (Lawless, 2009). Content reusability techniques can be broadly classified into three main forms; namely reuse through *encapsulation, shared publishing* and *modification* (Levacher, 2014).

2.6.1 Content Encapsulation

Content encapsulation involves the production of resources according to standard content models and formats. Fundamental objectives of content encapsulation are the easy portability of content items from one application to another as well as the reusability of this content (Bohl et al., 2002). This technique aims to make content resources available in large resource repositories to improve their interoperability across content consumers. The field of eLearning provides a good example of content encapsulation techniques (Mödritscher et al., 2004). This field utilised content encapsulation to improve the reuse and interoperability of learning resources produced across educational institutions where related content resources are grouped together and modelled into aggregates called Learning Objects (LOs). LOs are digital, self-contained 'chunks' of learning content (Wiley, 2000) that aim to enable content reuse outside the context in which it was created and dynamic, 'on the fly' sequencing of resources (Tasso et al., 2014). Examples of modelling standards, among others, are: IEEE Learning Object Metadata (LOM)[10], Dublin Core[11], IMS Learning Resource Metadata[12], Sharable Content Object Reference Model (SCORM)[13] and Grid Learning Object Repository (G-LOREP) (Pallottelli et al., 2010). These modelling standards have been introduced to support the creation of precise definitions of individual resources with information regarding how these resources can be reused within different educational institutions (Tasso et al., 2018).

Different eLearning systems have relied upon using such modelling standards as a metadata standard for their content model. (Conlan et al., 2002) proposed the APeLS system that separated the learning content from the adaptive linking logic or narrative, which improved the possibilities of reusing a piece of learning content. They use the IMS Learning Resource Metadata as the basis for the content model schema in APeLS in order to describe both technical and pedagogical aspects of the LO. This descriptive metadata information in APeLS allows course designers to easily discover learning content in the content repository by providing appropriate descriptive metadata. Furthermore, it can be used by an adaptive engine to select appropriate content where there may be many candidate LOs available to fulfil a learning or technical requirement.

[10] https://standards.ieee.org/ [Accessed: March 18, 2018]
[11] http://dublincore.org/ [Accessed: March 18, 2018]
[12] https://www.imsglobal.org/metadata/index.html/ [Accessed: March 18, 2018]
[13] https://scorm.com/ [Accessed: March 18, 2018]

(Farrell et al., 2004) proposed an eLearning system that uses three modelling standards as the basis for content interoperability with other applications, namely: IEEE LOM, IMS Content Packaging[14], and W3C Resource Description Framework (RDF)[15]. LOM provides an information model that defines the structure of a metadata instance for a learning object. A metadata instance describes relevant characteristics of the learning objects grouped into categories such as[16]: general (e.g. identifier, title), educational (e.g. instructional role, typical learning time) and classification (e.g. topic). The IMS Content Packaging Specification provides the functionality to describe and package learning materials, into interoperable, distributable packages and addresses the description, structure, and location of online learning materials. Learning objects are connected into coherent paths based on their LOM topic classifications and the proximity of these topics in a RDF graph. This graph includes nodes for topics and edges for topic relationships that are encoded as RDF entities and properties, respectively. Using these modelling standards, along with an instructional sequencing policy makes the system capable of arranging the learning objects on a path that suits a particular learning sequence.

(Savić et al., 2018) proposed a course management system which stores a course model represented as distinct machine-readable components containing domain knowledge of different course aspects. In their system, they have built their own ontology of learning objectives that relies on Bloom's revised taxonomy (Anderson et al., 2001) where a learning objective refers to the representation of domain knowledge that should be mastered during a specific course. Each learning objective is mapped (using the ontology) to one or more learning resources where a learning resource is any digital content that can be used for the achievement or evaluation of a learning objective in a course. They use the IMS Content Package specification for describing learning resources using metadata. This metadata information is stored in their model using a separate component in the system which maps learning resources to the ontology of learning objectives. Using this information (about learning objectives and learning resources), the system can export courses based on the formats required by a Learning Management System (LMS). These systems show that content encapsulation standards allow a common structure and descriptive vocabulary to be used across resource consumers, which supports the discovery

[14] https://www.imsglobal.org/content/packaging/index.html/ [Accessed: March 18, 2018]
[15] https://www.w3.org/RDF/ [Accessed: March 18, 2018]
[16] A metadata instance in IEEE LOM standard describes relevant characteristics of the learning objects grouped into general, life cycle, meta-metadata, technical, educational, annotation, relation, rights, and classification categories. However, their system depends only upon the General, Educational, and Classification metadata.

and reuse of content resources from various origins by each individual application. However, as the volume of resources produced by an institution grows, the amount of manual labour required to describe and structure these resources also increases (Bailey et al., 2006). Furthermore, repositories produced by content encapsulation techniques can still be seen as "closed pools" of interoperable resources, since the resource publication and delivery mechanisms are specific to each repository.

2.6.2 Shared publishing

Approaches to reuse that use standard shared publishing have attempted to overcome the limitations encountered by encapsulation techniques. They did so by providing a common publishing mechanism across resource repositories. The Semantic Web initiative (Berners-Lee et al., 2001) is the most successful example of this type of content reuse. The semantic web is an extension of the WWW that was proposed by the World Wide Web Consortium (W3C) through standards that promote common data formats and exchange protocols on the Web. The main objective of the semantic web is to provide a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. The Linking Open Data project[17] became one of the main showcases for successful community-driven adoption of semantic web technologies (Feigenbaum et al., 2007).

While the architecture of the WWW enabled the ease of content publication by millions of authors, linked open data focuses upon improving the reuse of such content by describing it with machine readable data. It is mainly based upon three core open standard technologies, namely: i) Resource Description Framework (RDF) to encapsulate data relationships; ii) URIs to identify individual content resources; and iii) a standard transfer protocol (mainly HTTP) to retrieve RDF data associated with each single resource. RDF is a metadata model that allows content resources to be described by a statement about each individual resource in the form of subject-predicate-object declarations called triples, where URIs are used to represent each subject, predicate or object. The triples are saved in so-called *triple stores*, repositories such as: Mimir from GATE[18], Openlink's Virtuoso[19] and AllegroGraph[20]. These triples enable each individual resource or relation to be shared between machines, and hence enables relationships between resources to be

[17] https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData [Accessed: March 18, 2018]
[18] https://gate.ac.uk/mimir/ [Accessed: March 18, 2018]
[19] https://virtuoso.openlinksw.com/ [Accessed: March 18, 2018]
[20] https://franz.com/agraph/allegrograph/ [Accessed: March 18, 2018]

reused across repositories (Zouaq et al., 2017). Moreover, ontologies can be used as the controlled vocabulary and semantic backbone between repositories for publishing data. (Kobilarov et al., 2009), for example, used Linked Data[21], MusicBrainz[22] and DBpedia[23] to integrate data and link documents across the different content repositories of BBC[24] domains, e.g. food, music, news etc.

In essence, content reuse through shared publishing removes the need for individual applications to modify its content consumption mechanism to each individual repository. Furthermore, it improves the accessibility of content resources from the different repositories as if they are from a single unique repository. However, both encapsulation and shared publishing reuse mechanisms do not involve the modification of the original resources themselves. Although reusing a content resource without modifying it can potentially include reusing it for purposes not originally planned by its author, the inability to modify a content resource limits the range of purposes for which it could be reused.

2.6.3 Content Modification

Due to limitations encountered in both encapsulation and shared publishing reuse mechanisms, other approaches were developed which modify the original resources in order to increase their amenability for reuse (Meyer et al., 2011). (Gonzalez-Barahona et al., 2006) proposed the Edukalibre platform to support the creation of collaboratively constructed educational materials. It is a collaborative system that provides version control management and conversion tools to produce several formats for each document in the system. Edukalibre takes inspiration from lessons learned in the open software development community.

Although Edukalibre facilitates collaboration between both educational practitioners and students to create and distribute open educational content, the modification of a content resource is performed manually using word processor software (e.g. OpenOffice[25]). Although this approach used the manual modification of content resources, automated alternatives have also been proposed. The Artequakt[26] system (Millard et al., 2003), for example, automatically extracts parts of content resources to create dynamic biographies of

[21] http://linkeddata.org/ [Accessed: March 18, 2018]
[22] https://musicbrainz.org/ [Accessed: March 18, 2018]
[23] http://wiki.dbpedia.org/ [Accessed: March 18, 2018]
[24] http://www.bbc.com/ [Accessed: March 18, 2018]
[25] http://www.openoffice.org/ [Accessed: March 19, 2018]
[26] Initially it was written as "Artequakt" in (Millard et al., 2003) and then written as "ArtEquAKT" in (Weal et al., 2007)

artists from content available on the web. In this example, all of the content presented to users originates from various fragments extracted from separately authored standalone documents.

(Ahmadi and Kong, 2008) followed the same technique to automatically adapt the desktop presentation of a content resource (web page) to a mobile presentation. Their approach relies on the DOM structure and the visual layout to divide the original Web page into several subpages, where each subpage includes closely related content and is suitable for display on the small screen. (Levacher et al., 2014) applies a web page fragmentation approach to automatically identify content fragments (slices) in web resources to repurpose them according to a request sent by an arbitrary content consuming application.

Although the reuse of a resource through modification can potentially include reusing it for purposes not originally planned by its author, relying only on the original structure of the content resource (DOM structure or paragraphs) to modify it however, certainly limits the range of purposes which it could be reused for. In other words, the structure of a content resource reflects the needs and point of view of the content author, which in turn does not reflect the needs and goals of individual applications. Furthermore, since these approaches modify content resources and produce segments before a request is done, the produced segments are considered static content items which in turn restricts the potential scenarios where such content items can be reused in. This in turn makes such approaches limited in responding to the different potential forms of requests.

Hence, there is a need for an approach that can modify the content resource in a manner that produces content fragments at the lowest level of granularity, regardless of the original structure of such resource. Furthermore, such an approach should allow the creation of a content fragment on-the-fly according to the characteristics of an arbitrary request. The approach should also provide a generic content publication and delivery mechanism to allow content consumers to easily acquire content items without the need to adjust their content acquisition mechanism. In order to build such approach, content needs to be more understandable. Hence, the research in this thesis uses NLP techniques to understand content and thus enhance its discoverability and reusability.

2.7 Natural Language Processing in Adaptive Systems

The main objective sought by adaptive systems is to address the challenge of producing adaptive compositions from different information sources in order to deliver content in a

form that is most suitable to an individual user. As outlined in the preceding sections of this chapter, adaptive systems which attempt to repurpose and reuse open corpus content are limited by a reliance only upon the original structure of the content that reflects the needs and the perspective of the author of the resource. While each individual application has its own requirements (based on its users), relying on this structure does not necessarily reflect these requirements. Furthermore, these systems are limited in that they do not deeply "understand" the content, which in turn limits their capabilities to supply appropriate content for use in defined contexts.

As a result, a number of studies have applied Natural Language Processing (NLP) techniques to understand content and enhance its discoverability and reusability (Alfonseca et al., 2007; Leoncini et al., 2012). The field of NLP aims to gather knowledge on how human beings understand and use language so that appropriate tools and techniques can be developed to make computer systems "understand" and manipulate natural languages to perform a range of desired tasks (Chowdhury, 2003).

ArtEquAKT (Millard et al., 2003; Weal et al., 2007) utilises an Information Extraction (IE) (section 2.2.3) approach that automatically extracts factual information items together with sentences and paragraphs from unstructured web documents to create dynamic biographies of artists. The authors proposed a Relation Extraction approach (e.g. (Aitken, 2002)) to extract pre-defined relation types between two identified entities.

(Sathiyamurthy & Geetha, 2011) used a text segmentation algorithm to segment technical documents in the computer science domain for the purpose of eLearning. They use the TextTiling (Hearst, 1994) (section 2.3.5) algorithm along with domain and pedagogical ontologies to apply block level text segmentation for eLearning material. In the segmentation process, a topic from the domain ontology is assigned to each block in a given document. If consecutive blocks have the same topic all blocks are combined together to form a single segment. If adjacent blocks differ, with different topics or with different cue-words from the pedagogical vocabulary, then they are separated into disparate segment. The segments produced are then used as content items for building eLearning courses.

(Beck et al., 2014) also investigated how text segmentation algorithms can be applied to automatically transform unstructured text into coherent pieces appropriate for generating eLearning courses. The main objective of using text segmentation is to provide eLearning course designers with a tool to efficiently organize existing textual content for new

eLearning courses. Their approach uses Wikipedia as the source for eLearning material and produces two levels of content items, macro and micro levels. The macro level corresponds to sections from different articles in Wikipedia, and the corresponding micro structure consists of subsequent paragraphs from these sections. They used two different text segmentation algorithms: the linear segmenter TopicTiling (Riedl & Biemann, 2012) for the macro level and the hierarchical segmenter BayesSeg (Eisenstein & Barzilay, 2008) (section 2.3) for the micro level. Their intuition behind using a hierarchical segmenter in the micro level is that the length of the produced Knowledge Objects (KO) in that level should be adapted to the intended skill and background of the learners. This assumption, in fact, aligns with the research in this thesis. The intuition behind using hierarchical text segmentation in processing content in this thesis, is that it enables the production of different levels of content granularity. This in turn makes content more flexible and enables the production of different compositions of content items to meet the adaptation requirements of individual users or applications.

(Alfonseca et al., 2007) proposed the WELKIN system that relies on various NLP techniques to build adaptive web sites. The system comprises two processing steps, off-line and on-line. The off-line processing step analyses the source text before the user interacts with the system. During this step, the domain-specific texts provided by the user are processed and analysed with some linguistic tools. These tools include: a tokenizer, a sentence splitter, a stemmer, a part-of-speech tagger, several partial parsers for Noun Phrases and Verb Phrases, and a module that identifies text sections and chapters. After this linguistic processing, a term extraction approach is used to locate the different entities in text such as dates, scientific names, etc. On the other hand, the on-line processing step is performed whenever a user interacts with the system. In this step, the system starts to compose and generate a website from the processed content in the off-line step. This content is presented based on the amount of information the user is willing to read where the user can indicate this preference in different ways, such as the total number of words that the generated website must contain; a fixed compression rate to be performed to all the web pages; or the amount of time that they want to spend reading the whole site. Based on one of these preferences, the system uses a sentence extraction procedure based on genetic algorithms (Alfonseca & Rodríguez, 2003) to summarise the generated pages according to the user's preference.

(Leoncini et al., 2012) proposed a semantic-based framework for summarisation and page segmentation. The framework uses text summarisation to extract a concise summary from

a web resource, which outlines the relevant topics addressed by the textual data, thus discarding uninformative, irrelevant contents. It also applies web page segmentation to generate segments that point out the relevant text parts of the resource. The first step in the framework is processing the raw text to identify individual words and sentences. Concepts are subsequently assigned to each word, using EuroWordNet synsets[27] (Vossen, 1998) and then grouped into domains. After identifying the set of domains addressed in the web page, text summarisation is then obtained by detecting in the original textual source the sentences that are most highly correlated to the domains included in the identified set. These sentences are then ranked according to the single terms they involve. This ranking is further used to select the portions of the web page that deal with the main topic addressed by the user. The summarisation approach used in this framework can produce two types of summaries: 1) a summary that describes the overall content of the web page, and therefore does not distinguish the various domains included in that page and 2) multiplicity of summaries, one for each domain addressed in the page.

These systems have tried to utilise NLP techniques to structure content and then use this structure to support the use of this content in adaptive systems. However, except for (Alfonseca et al., 2007), they did not provide new methods to enhance these NLP techniques. Furthermore, it appeared to the author's knowledge that there has not been a significant volume of work carried out in this area. Hence, the key motivation of this research is to examine different methods to enhance NLP techniques in order to use them to mine textual content for content adaptation. The research in this thesis mainly focuses on the use of text segmentation to enhance content discoverability and reusability for content consuming applications.

2.8 Chapter Summary

This chapter presented an overview of Natural Language Processing (NLP) techniques and presented a state of the art review of the existing approaches for text segmentation as a technique for structuring textual content. It first reviewed the different criteria that the text segmentation task is categorised according to. From a text representation perspective, text segmentation approaches were categorised into linear and hierarchical approaches. Reviewing linear segmentation approaches identified that they can only produce a single-level segmentation of a document. However, considering the structure of a document as

[27] http://projects.illc.uva.nl/EuroWordNet/

a sequence of segments is in certain discord with most theories of textual content structure, where it is more usual to consider documents as trees. Thus, hierarchical text segmentation is seen as a method that can effectively represent a document as a tree-like hierarchy structure.

The chapter presented a focused review of hierarchical text segmentation approaches and how they process text. The review showed that these approaches are limited by the fact that they can only process the information that they can 'see'. In other words, they are based on the lexical and/or syntactic representation of text, a method that relies mainly upon the traditional bag-of-words representation of text to measure similarity (or dissimilarity) between text blocks. However, a representation based solely on the endogenous knowledge in the documents themselves does not reveal much about the meaning of the text.

Building on the review and analysis of the state of the art approaches to text segmentation, the next chapters (Chapter 3 and Chapter 4) present two novel approaches to hierarchical text segmentation that utilise external knowledge resources in order to enrich text and infer more information about text constituents.

The chapter also presented an overview of adaptive systems, as an application for content adaptation, and reviewed their anatomy, their models and in particular their content model. Closed and open corpus content models were reviewed in order to better illustrate how adaptive systems process different types of content. The chapter then presented a review on different approaches utilised by adaptive systems to discover content according to their users' needs. Content reusability techniques were also reviewed along with their limitations. Furthermore, a review of current NLP techniques utilised by adaptive systems was undertaken. The aim of this review is to investigate how adaptive systems use NLP techniques in processing content resources and how these techniques contribute to the provision of adaptive experiences to adaptive systems' users.

Building on the review and analysis of adaptive systems and content discoverability and reusability techniques, Chapter 5 presents a content-supply service (named CROCC) that facilitates the use of the new segmentation approach (Chapter 4) for content discoverability and reusability for adaptive systems. Additionally, Chapter 6 presents a user-based evaluation of the effectiveness of the proposed service in content discoverability and reusability.

3. OntoSeg: A Novel Approach to Text Segmentation using Ontological Similarity

3.1 Introduction

As outlined in Chapter 2, many adaptive systems have relied upon the original structure of content resources (HTML structure or paragraph structure) to produce content fragments and hence use them in content adaptation. Since this structure does not necessarily reflect the needs or preferences of individual users or applications, more recent systems have tried to employ text segmentation techniques in order to build a structure out of content resources based on the text itself, rather than the structure provided by the content author (section 2.7).

Text segmentation is the process of placing boundaries within text to create segments according to some task-dependent criterion. It is considered an essential task for various NLP tasks (Beck et al., 2014; Bokaei et al., 2016). Text segmentation aims to divide text into coherent segments which reflect the sub-topic structure of the text. As outlined in Chapter 2, current approaches to text segmentation are similar in they all use the traditional word-frequency metrics to measure the similarity between two regions of text, so that a document is segmented based on the lexical cohesion between its words (section 2.3.6). However, the relationship between segments may be semantic, rather than lexical or syntactic.

Various NLP tasks are now moving towards the semantic web and the use of ontologies. In Information Retrieval, for example, systems that are based on keywords provide limited capabilities to capture the topical interests of users and topics contained within content. In order to solve these limitations, the idea of semantic search, based on the semantic meaning of text, has been the focus of a wide body of research and many ontology-based IR systems have been developed (Fernández et al., 2011; Meštrović and Calì, 2017; Selvalakshmi and Subramaniam, 2018). Hence, a need for segmenting and representing text based on the semantic (ontological) similarity between its constituents arises.

This chapter proposes OntoSeg (Bayomi et al., 2015), a novel approach to hierarchical text segmentation based on the semantic similarity between text blocks. In contrast to traditional text segmentation approaches that rely upon bag-of-words representation of content, OntoSeg uses semantic similarity to explore conceptual relations between text

segments and a Hierarchical Agglomerative Clustering (HAC) algorithm to represent the text as a tree-like hierarchy that is conceptually structured. The output is a hierarchical structure of the underlying content that is constructed based on how conceptually similar text blocks (one or more sentences) are to each other.

The aim of this chapter is to answer the first research question posed by this thesis (section 1.2):

> *To what extent can the semantic representation of unstructured textual content be exploited by novel text segmentation approaches to build a document structure?*

and to contribute to its second objective (RO 2). The architecture of OntoSeg is presented and a set of experiments are described, which have been carried out in order to evaluate the performance of OntoSeg using a well-known evaluation metrics. The evaluation comprises different experiments where each experiment evaluates OntoSeg from a different perspective. Experiments demonstrate that segmenting text based on the semantic similarity is applicable with a low error rate. The performance of OntoSeg is also compared against a set of state of the art approaches using a dataset widely used in the literature.

3.2 OntoSeg Architecture

The architecture of OntoSeg consists of three phases:

1- Semantic annotation.
2- Calculating similarity between text blocks (sentences or paragraphs).
3- Hierarchical Agglomerative Clustering (HAC).

3.2.1 Semantic annotation

In this phase, the text is semantically annotated using a named entity recognition algorithm (section 2.2.3) and text entities are extracted. Each entity is then mapped to its class or classes in an ontology and the text is represented as a sentence-based vector of classes. This vector is then used as an input to the following phase.

A large number of ontologies now exist, some of which are domain-specific (such as the *MeSH*[1] ontology of medical and biomedical terms), while others are cross-domain such as *DBpedia*[2] (Auer et al., 2007). As the research in this thesis is not focusing on a specific domain, the DBpedia ontology is used as the underlying knowledge base, as opposed to

[1] http://www.nlm.nih.gov/mesh
[2] http://dbpedia.org/

a domain-specific alternative. *DBpedia Spotlight*[3] (Daiber et al., 2013; Mendes et al., 2011) is used as the named entity recognition system to extract entities from the targeted text. DBpedia Spotlight is a tool for automatically annotating mentions of DBpedia resources in text, providing a solution for linking unstructured information sources to the Linked Open Data cloud[4] through DBpedia. DBpedia Spotlight recognises entities that have been mentioned in text and subsequently matches these entities to their classes in the DBpedia ontology. For each annotated entity in the text, the classes that match this entity are extracted. For example, BARACK OBAMA, as an entity, matches with the DBpedia classes: *["Politician", "Person", "Agent"]*. Since the elementary blocks for the proposed approach are sentences (see section 3.4.2), each sentence in the text is represented as a vector of entities, and each entity is represented by a set of classes that match the entity from DBpedia. A sentence-based vector is built and a similarity between its adjacent vectors is measured as discussed in the following subsection.

An example of three sentences annotated by DBpedia Spotlight is depicted in Figure 3.1. The underlined words in this figure represent the extracted entities in each sentence[5]. Each entity is then mapped to its class or classes in the DBpedia ontology. After that each sentence is represented as a vector of entities where each element in that vector is represented as a set of classes. Figure 3.2 depicts how the three sentences in Figure 3.1 are represented as vectors of sets of classes.

Donald Trump is now the president of the United States.
Barack Obama was the president before him.
Michael Jackson was a famous singer.

Figure 3.1 Example of three sentences annotated by DBpedia Spotlight

```
[["President","Politician","Person","Agent"],["Place","Country"]]
[["Politician","Person","Agent"]]
[["Person","MusicalArtist","Artist"]]
```

Figure 3.2 A vector representation of the three sentences after mapping entities to their classes from DBpedia ontology

[3] https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki
[4] http://lod-cloud.net/
[5] DBpedia Spotlight requires a disambiguation confidence parameter that can be set from 0 to 1. The value of the confidence parameter in the example shown in Figure 3.1 (and in all work in this chapter) was set to 0.5.

3.2.2 Similarity Computation

The key idea proposed in OntoSeg is to use the semantic similarity between text blocks in the segmentation of text. A text block is the elementary unit considered by the segmentation algorithm, which could be one sentence or multiple sentences (a paragraph).

The similarity between text units is calculated based on two similarity measures: (1) Ontological[6] (semantic) similarity and (2) Lexical similarity.

3.2.2.1 Ontological Similarity

Ontological similarity relates to identifying conceptually similar, but not necessarily lexically similar, terms. For example, in Figure 3.1, Donald Trump and Barack Obama are not lexically similar. However, if we consider the classes that both names belong to (i.e. *["Politician", "Person" "Agent"]* in Figure 3.2), it is reasonable to say that the two entities are conceptually similar to each other. This is the main idea behind OntoSeg; in contrast to a lexical interpretation of text, OntoSeg interprets text based on its semantic meaning and uses ontological similarity as a means of measuring how similar two adjacent sentences are to each other.

Ontological similarity has been widely used in many research fields such as: (1) *Information Retrieval*, to improve accuracy of current retrieval techniques and for semantic indexing (Meštrović and Calì, 2017); (2) NLP tasks, such as word sense disambiguation (Prokofyev et al., 2013), synonym detection (Chaves-González and Martínez-Gil, 2013) and sentiment analysis (Cambria et al., 2015); (3) *Knowledge management* tasks such as thesauri generation (Curran, 2002), information extraction (Shah and Jain, 2014), semantic annotation (Sánchez et al., 2011) and ontology merging and learning (Priya and Kumar, 2015), in which new concepts should be discovered or acquired from text in order to relate them to already existing concepts.

Ontology-based similarity can be classified into three main approaches (Elavarasi et al., 2014):

1- *Edge-counting approaches*: where the minimum path length connecting two corresponding ontological nodes via *is-a* links is used as a straightforward method to calculate the similarity between the concepts represented by those nodes (Wu and Palmer, 1994; Gao et al., 2015). In ontology structure, the *is-a* relations group the

[6] Ontological similarity refers to the Semantic similarity based on an ontology. Both phrases are used interchangeably in this chapter.

classes according to how they are conceptually related to each other. Given a pair of classes, c1 and c2, a well-known method with intuitive explicitness for assessing their similarity is to calculate the distance between these classes in an ontology hierarchy; the shorter the distance, the higher the similarity. In the case that multiple paths between the nodes exist, the shortest distance of all paths is used.

2- *Feature-based approaches*: contrary to edge-counting approaches, feature-based approaches assess the similarity between concepts as a function of their properties (Jiang et al., 2015). They take into account common and noncommon features of the compared concepts.
3- *Information Content (IC) based approaches*: these approaches are associated with the probability of appearance of each concept in the taxonomy, computed from their occurrences in a given corpus (Jiang et al., 2017). IC of a term is computed according to the negative log of its probability of occurrence. In this manner, infrequent words are considered more informative than common ones.

In this research, the Edge-counting approach proposed by Wu and Palmer (Wu and Palmer, 1994) is used as its performance has been shown to be better than the other methods (Lin, 1998; Hill et al., 2015). The principle behind this approach is that the similarity of two concepts is defined by how closely they are related in the hierarchy, i.e., their structural relations. Given two concepts c_1 and c_2 the conceptual similarity $ConSim$ between them is:

$$ConSim(c_1, c_2) = 2 * \frac{N}{N1 + N2} \qquad 3.1$$

where N is the distance between the closest common ancestor (CS) of c_1 and c_2 and the ontology root, and $N1$ and $N2$ are the distances between the ontology root on one hand and $C1$ and $C2$ on the other hand respectively. Figure 3.3 shows how the similarity between two concepts in an ontology is measured.

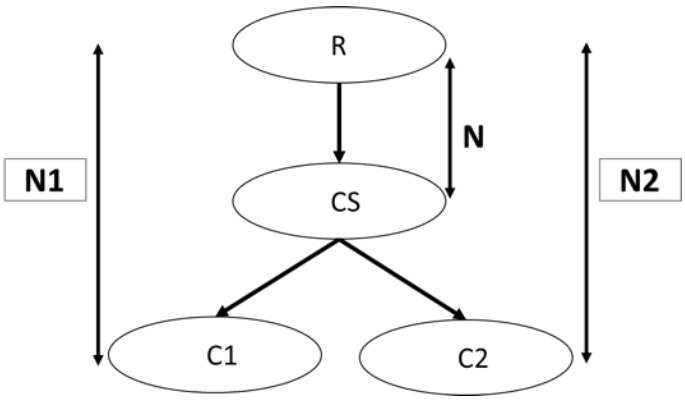


Figure 3.3 Example of ontology extract (Slimani et al.,2006)

Accordingly, the similarity between two entities can be defined as a summation of weighted similarities between pairs of classes in each of the entities. Given two entities *E1* and *E2*, the similarity $EntSim$ between them is:

$$\boldsymbol{EntSim(E1, E2)} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \boldsymbol{ConSim}(c_i, c_j)}{\boldsymbol{m \times n}} \qquad 3.2$$

where *m* and *n* are the two sets of classes that *E1* and *E2* have respectively.

Equation (3.2) calculates the similarity between two entities, where each entity belongs to one or more classes. For example, BARACK OBAMA as an entity is mapped to three DBpedia classes: *["Politician", "Person" "Agent"]*, and DONALD TRUMP is mapped to four DBpedia classes: *["President", "Politician", "Person" "Agent"]*. Hence, although the two entities are not lexically similar, they are deemed ontologically similar. This is the idea behind ontological similarity: it measures the similarity between entities according to the conceptual characteristics which they share. As another example of how ontological similarity differentiates between entities, consider MICHAEL JACKSON as an entity that is mapped to four DBpedia classes: *["Person", "MusicalArtist", "Artist"]* (Figure 3.2). Intuitively, the two entities BARACK OBAMA and DONALD TRUMP are more ontologically similar to each other than either of them is to MICHAEL JACKSON.

On a text-block level (a sentence for example), the similarity between two blocks can be defined as the summation of weighted similarities between pairs of entities in each of the units. Given two text blocks *B1* and *B2*, which have a set of entities *a* and *b* respectively, the similarity *BlockSim* between *B1* and *B2* is:

$$BlockSim(B1, B2) = \frac{\sum_{i=1}^{a} \sum_{j=1}^{b} EntSim(E_i, E_j)}{a \times b} \quad 3.3$$

where *a* and *b* are the two sets of entities that *B1* and *B1* have respectively.

3.2.2.2 Lexical similarity

Lexical similarity has been widely used in the literature in text segmentation (Hearst, 1994; Choi, 2000; Tsunoo et al., 2017; Wang et al., 2017), and as its name suggests, it splits text into segments that are lexically coherent. Lexical cohesion refers to the connectivity between two portions of text in terms of word relationships. Although text blocks might share ontological similarities between each other, it may be the case that ontological similarity alone is not sufficient to measure how text blocks are coherent with each other. The reasons for this include:

1- Text blocks may not contain any entities.
2- The entity extraction algorithm may not discover some entities in the text block.
3- The extracted entities from a text block may not be sufficient to reflect the similarity between text blocks.
4- The ontology being used may not cover all the text mentions.

Thus, the lexical overlap between text blocks should be part of the overall similarity measure. As a result, the similarity measure is enriched by obtaining the lexical similarity between text blocks and combining it with the ontological similarity. To measure the lexical similarity between text blocks, first, stop words are removed from the text as they are generally assumed to be of little, or no, informational value. Then the remaining words are stemmed (section 2.2.2) using Porter stemmer (Porter, 1980) and each block is represented by a lexical frequency vector. For each adjacent text blocks, a lexical vector cosine similarity is calculated. It is defined as the cosine of the angle between two vectors *v* and *w* such that:

$$cos(v, w) = \frac{\vec{v}.\vec{w}}{\|\vec{v}\|.\|\vec{w}\|} \quad 3.4$$

3.2.3 Hierarchical Agglomerative Clustering (HAC)

Hierarchical clustering algorithms have been studied extensively in the clustering literature (Jain and Dubes, 1988). The general concept of agglomerative clustering is to successively merge documents into clusters based on their similarity with one another. The agglomerative clustering technique could be transferred from document level into text level, where the clustering process is done between text blocks, within a document (as opposed to across whole documents) (Yaari, 1997; Wang et al., 2017). When applying Hierarchical Agglomerative Clustering on text blocks the algorithm successively agglomerates blocks that are coherent to each other, thus forming a text structure.

The idea behind using HAC in text segmentation is that it is a bottom-up clustering approach, which means that it starts from the smallest chunks (one sentence in OntoSeg) and then builds the text hierarchy by merging text blocks (clusters) based on how near or similar they are to each other. In contrast, the top-down (divisive) clustering approach starts from the full document and then divides the text into smaller blocks based on how far (i.e. how different) they are from each other. Hence, the output of the bottom-up approach can be regarded as hierarchically coherent tree. Thus, the method of Hierarchical Agglomerative Clustering for text is useful to support a variety of search methods because it naturally converts text into a tree-like hierarchy and provides different levels of granularity for the underlying content; this can then easily be leveraged for the content discoverability process.

Unlike general HAC for clustering documents, where at each stage the proximity of the newly merged object to all other available objects is computed, at the text level we compute only the similarity of the text block to its two neighbours. This is because the linear order in the text is required to be preserved in the structure. The implication on complexity is that while the general HAC algorithm for documents takes an order of $O(N^2)$ steps, it takes only $O(N)$ when used at the text level.

The algorithm successively clusters "coherent" segments based on the accumulation between the ontological and lexical similarity scores between text blocks, which guarantees the ontological and lexical cohesion between agglomerated segments. The HAC algorithm for text segmentation, based on blocks as the elementary segments, is shown in Figure 3.4.

Input: *Ontological and Lexical representations of text*
Output: *a hierarchical structure of the document*
begin
while (number of blocks>1) do
foreach *block Bi* **do**
calculate similarity between B_i *and its neighbors*
Ontological Similarity: Osim (B_i, B_{i+1}) *and Osim* (B_i, B_{i-1})
Lexical Similarity: Lsim (B_i, B_{i+1}) *and Lsim* (B_i, B_{i-1})
Total Similarity*:*
Tsim (B_i, B_{i+1}) = *Osim* (B_i, B_{i+1}) + *Lsim* (B_i, B_{i+1})
Tsim (B_i, B_{i-1}) = *Osim* (B_i, B_{i-1}) + *Lsim* (B_i, B_{i-1})
if **(***sim*(B_i, B_{i+1}) >= *sim*(B_i, B_{i-1})**) then**
merge (B_i, B_{i+1}**)**
end
else
merge (B_i, B_{i-1}**)**
end
end
end
end

Figure 3.4 OntoSeg Algorithm

Conceptually, the process of agglomerating blocks into successively higher levels of clusters creates a cluster hierarchy (dendrogram) for which the leaf nodes correspond to individual blocks (sentences in OntoSeg), and the internal nodes correspond to the merged groups of clusters. When two groups are merged, a new node is created in this tree corresponding to this larger merged group. The two children of this node correspond to the two groups of blocks which have been merged to it. Figure 3.5 shows the resulted dendrogram from the algorithm for a sample text with one sentence as block size.

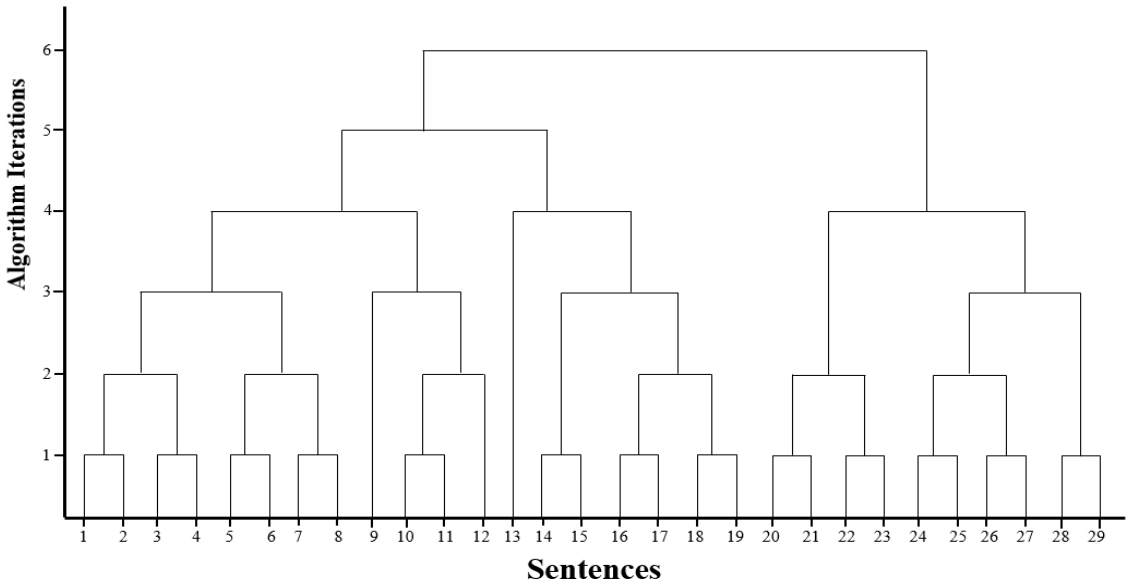


Figure 3.5 Sentences dendrogram of a sample text

3.3 From Hierarchical into Linear Representation

OntoSeg produces a tree that can be used as a visual illustration of the underlying hierarchical structure of a document. Figure 3.6 depicts a tree representation of a sample text of 10 sentences. The benefit of this tree is that it represents different levels of granularity of the document, which in turn means that the document can be segmented into different segmentation levels. This is a powerful criterion in the hierarchical representation of text. In contrast to linear representation, in each level of the structure (tree), segmentation with different levels of details can be obtained and can be usefully applied to many other tasks' needs.

In order to convert a hierarchical representation into a linear representation a threshold corresponding to the number of the segments needed is set and the level that contains the corresponding number of nodes in the tree is extracted. For example, in Figure 3.6, if the specified threshold is 2, the level under the root is selected as it has two segments.

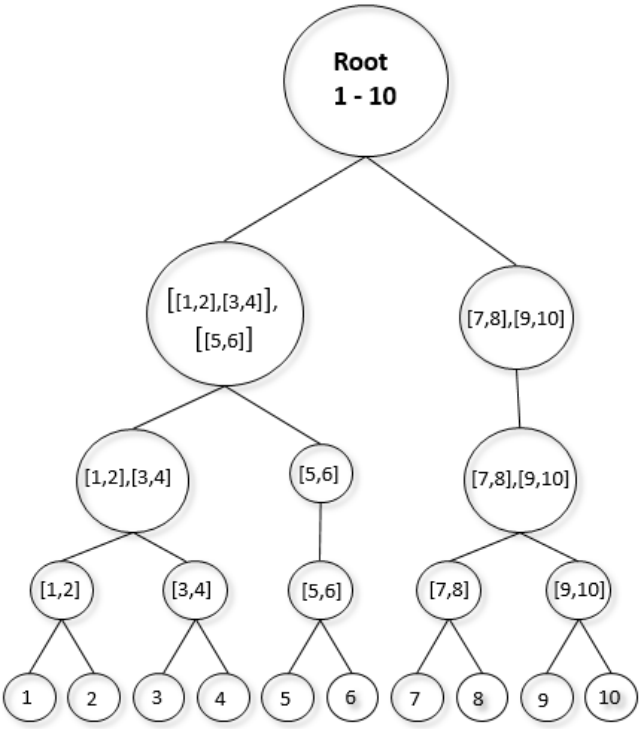


Figure 3.6 A tree representation for a text from 10 sentences

If this threshold number is not represented in one of the tree levels, a flattening process is applied to the largest nodes. For example, suppose that the specified number is four segments, and in one of the tree levels the number of nodes (segments) is three nodes. As now we need one more segment, the largest node (largest in terms of child nodes) is flattened by obtaining the two subsequent nodes that constitute this large node, i.e. we go down a level in the tree for this large segment. This method of flattening the tree guarantees that the coherency between the obtained segments is preserved.

3.4 Evaluation

3.4.1 Experimental Setup

The output from OntoSeg is a tree that represents the text hierarchy. As depicted in Figure 3.6 each level in the tree represents a level of granularity for the text where each node, in that level, represents a segment that contains coherent blocks. As mentioned before, a linear representation of text can be obtained from such a tree, which means that OntoSeg can be evaluated as a linear text segmentation approach[7].

[7] It is worth mentioning that at the time of evaluating OntoSeg there was no hierarchical text segmentation dataset publicly available. However, in the evaluation of C-HTS (Chapter 4), two hierarchical text segmentation datasets are used where the performance of both OntoSeg and C-HTS is assessed.

In this experiment, the efficiency of OntoSeg is evaluated on Choi's dataset[8] (Choi, 2000). This dataset has been widely used in linear text segmentation evaluation (Riedl & Biemann, 2012a; Du et al., 2013; John et al., 2017). The dataset consists of documents made up of ten concatenated text segments. Each segment consists of the first *n* sentences of a randomly selected document from the Brown Corpus (Francis, 1964). The dataset is divided into four subsets and are listed in Table 3.1. There is a total of 700 text documents in the dataset.

Table 3.1 Choi's dataset statistics

Range of *n*	3-11	3-5	6-8	9-11
# samples	400	100	100	100

Each document in the dataset is processed and two vectors are generated: the ontological and the lexical. Since the elementary text blocks in OntoSeg consist of sentences, each sentence in the ontological vector is represented as a vector of sets of DBpedia classes where each set represents an entity that is extracted from the sentence. These sets of classes are used to measure the ontological similarity between sentence vectors according to equations (3.1), (3.2) and (3.3). To build the lexical vector, first the stopwords[9] are removed from the text and then the remaining terms are stemmed (using Porter stemmer); after this, each sentence is subsequently represented as a term-frequency vector. The lexical similarity between vectors of adjacent sentences is then determined by calculating the cosine similarity between them as in equation (3.4).

A HAC algorithm is then applied on the obtained vectors. For the ontological vector, the ontological similarity score is calculated between each vector and its two neighbours (section 3.2.2.1). A lexical similarity score is also obtained for the lexical vectors (section 3.2.2.2). The final similarity score between two adjacent sentences is the combination of their ontological similarity and lexical similarity scores. For each set of three neighbouring sentences, the middle sentence is merged with the one that is most similar to it. For example, if the three sentences are denoted as *A*, *B* and *C*. Sentence *B* is merged with *C* if the similarity score between *B* and *C* is higher than the score between *A* and *B*.

When the two neighbours are merged together they form a new text block (cluster) and two new vectors (ontological and lexical) are defined based on the new block to be used in the next iteration of the algorithm. Iteratively, the algorithm applies the same process

[8] Choi's C99 release and the dataset are available here: https://github.com/logological/C99 [Accessed: May 03, 2018]
[9] Since Choi's dataset is not focused on a specific domain, a general domain stopwords list for English was used.

between adjacent blocks until it merges all text blocks into one single cluster and a tree representation of the text is produced. A linear segmentation is then produced as described in section 3.3 where the threshold is set to 10 as each document in Choi's dataset consists of 10 segments.

3.4.2 Elementary Units for OntoSeg

The size of the elementary text blocks is considered a critical step in the segmentation process. (Yaari, 1997) used paragraphs as the elementary blocks for his segmentation algorithm and affirms that the size of a paragraph, as opposed to a sentence, contains sufficient lexical information for the proximity test. Also (Hearst, 1994) measured the cosine similarity between text blocks where text blocks consist of a fixed number of sentences (window). As a result, the quality of the produced segments, using the ontological similarity only, or the combination between the ontological and lexical similarity, is examined using varying window sizes: from one to four sentences.

Since the main contribution of OntoSeg is to segment text based on the ontological similarity between its blocks, the quality of the produced segments is evaluated first based on ontological similarity only. After that, the impact of adding lexical similarity to the ontological similarity using different weights for the two similarity measures is examined.

According to these considerations, four experimental runs were conducted (in each run, varying window sizes are used (1 to 4)):

Experiment 1: in this run, ontological similarity only is used.

Experiment 2: in this run, the combination between ontological and lexical similarity scores is used with α = 0.3, where α is used to specify the weight of each of the two similarity measures. Let $Osim$ and $Lsim$ denote the ontological and the lexical similarity scores respectively; the final hybrid similarity score ($Hsim$) between two text blocks B1 and B2 is:

$$\boldsymbol{Hsim(B1, B2) = \alpha\, Lsim + (1 - \alpha)\, Osim} \qquad 3.5$$

Hence, α = 0.3 means that the lexical score weight is 0.3 and the ontological similarity score weight is 0.7.

Experiment 3: in this run, both similarity scores are treated equally, i.e. α = 0.5.

Experiment 4: in this run, lexical similarity is given a higher weight by setting α =0.7.

3.4.3 Evaluation Metrics

It is common to evaluate text segmentation systems in terms of the two commonly-used evaluation metrics, P_k (Beeferman et al., 1999) and *windowDiff* (WD) (Pevzner & Hearst, 2002). Both metrics are penalty measurement metrics, which means that lower scores indicate higher segmentation accuracy. P_k was proposed as a measure that expresses a probability of segmentation error. To calculate P_k, we take a window of fixed width k, which is usually set to half of the average segment length in the reference partition, and move it across the segmented text, at each step examining whether the hypothesized segmentation is correct about the separation (or not) of the two ends of the window. P_k metric is defined as:

$$\boldsymbol{P_k} = \sum_{\boldsymbol{1 \le i \le j \le k}} \boldsymbol{D_k(i,j)} \left(\boldsymbol{\delta_{ref}\,(i,j)} \oplus \boldsymbol{\delta_{hyp}\,(i,j)} \right) \tag{3.6}$$

where $\delta_{ref}\,(i,j)$ is an indicator function whose value is 1 if sentences i and j belong to the same reference segment and 0 otherwise. Similarly, $\delta_{hyp}\,(i,j)$ is 1 if the two sentences are hypothesized as belonging to the same segment and 0 otherwise. The $\oplus$ operator is the XOR operator. The function D_k is the distance probability distribution that uniformly concentrates all its mass on the sentences which have a distance of k.

windowDiff is stricter as it not only decides whether there is a mismatch between the hypothesized segment and the reference segment, it also counts the difference of the number of segment boundaries in the given window between the two segments. Thus, the results of *windowDiff* are generally higher than those of P_k. *windowDiff* is defined as:

$$\begin{aligned} &windowDiff\,(ref\,, hyp) \\ &\qquad = \frac{1}{K-k} \sum_{i=1}^{K-k} (|b(ref_i, ref_{i+k}) - b(hyp_i, hyp_{i+k})| > 0) \end{aligned} \tag{3.7}$$

where ref is the correct segmentation for reference, hyp is the segmentation produced by the model, K is the number of sentences in the text, k is the size of the sliding window and $b(i, j)$ is the number of boundaries between sentences i and j.

Since it has been argued in (Pevzner & Hearst, 2002) that P_k has some weaknesses, *windowDiff* is used as the evaluation metric in all experiments. However, the P_k metric is

used to evaluate the performance of OntoSeg against the state of the art approaches, as these approaches were evaluated based on the P_k metric in the relevant publications.

3.4.4 Results

Table 3.2 shows the results of experiment 1 (using ontological similarity only) while applying different window sizes, from 1 to 4 sentences per text block. From the results we can see that the error rates are not high for all the subsets (range from 0.15 to 0.32), which means that generating text segments based on the ontological similarity between its constituents is feasible, with low error rates. It can also be seen that varying the window size does not increase the quality of the segmentation; in contrast, it decreases the quality for some subsets.

Table 3.3 shows the results of experiments 2, 3, and 4 where the hybrid approach that combines the ontological and lexical similarities using different weights is evaluated.

Table 3.2 Ontological similarity error rates (*WD*) for different window sizes

Range of *n* / **Window**	**3-11**	**3-5**	**6-8**	**9-11**
W = 1	0.21	0.32	0.20	0.15
W = 2	0.21	0.32	0.21	0.15
W = 3	0.21	0.34	0.21	0.15
W = 4	0.22	0.34	0.21	0.15

The results of experiment 2 indicate that when α = 0.3, the error rates of the segmentation in all the subsets are less than the error rates using ontological similarity only (Table 3.2). In experiments 3 and 4, it can be noticed that as α increases (0.5 and 0.7 respectively), the error rates decrease. According to equation 3.5, when α increases, the lexical similarity weight is more than the ontological similarity weight. This means that, although generating text segments based on the ontological similarity is feasible, using it alone is not sufficient. In other words, using lexical similarity only ($\boldsymbol{\alpha}$ = 1 in equation 3.5) is better than using the semantic similarity in OntoSeg.

It is also noticed that, as in experiment 1, when the window size increases, the error rate also increases which means that the segmentation quality decreases.

Table 3.3 Hybrid approach error rates for different window sizes

Range of *n* / Window size	3-11	3-5	6-8	9-11
Experiment 2: α = 0.3				
W = 1	0.17	0.22	0.17	0.13
W = 2	0.19	0.29	0.18	0.14
W = 3	0.19	0.34	0.20	0.14
W = 4	0.20	0.33	0.20	0.15
Experiment 3: α = 0.5				
W = 1	0.16	0.21	0.16	0.12
W = 2	0.18	0.27	0.17	0.12
W = 3	0.19	0.33	0.19	0.13
W = 4	0.20	0.33	0.19	0.14
Experiment 4: α = 0.7				
W = 1	**0.15**	**0.19**	**0.15**	**0.11**
W = 2	0.17	0.25	0.16	0.12
W = 3	0.18	0.33	0.19	0.13
W = 4	0.20	0.33	0.20	0.14

3.4.4.1 OntoSeg Performance Against Other Approaches

To evaluate the quality of segmentations produced by OntoSeg, there is a need to compare its performance against the state of the art approaches based on the segmentation quality. An experiment was carried out where five different approaches were selected for this comparison. The five approaches are: TextTiling (Hearst, 1994), C99 (Choi, 2000), Segmenter (Kan et al., 1998), U00 (Utiyama and Isahara, 2001) and an approach proposed by (John et al., 2017) that segments text based on two similarity measures: lexical and semantic. For the lexical similarity, a vector is built from topics covered in each sentence. For the semantic similarity, a vector of verbs, nouns and the adjectives is built using a Part-of-Speech (POS) tagger where the similarity between two vectors is measured using the WordNet concept hierarchy. Both vectors (lexical and semantic) are combined to identify segment boundaries within the given text.

The intuition behind choosing these approaches is that they were evaluated on the same dataset used in this chapter (Choi, 2000). However, these approaches were evaluated using the P_k evaluation metric. Hence, in this experiment, OntoSeg is also evaluated with

the same metric. Table 3.4 presents a comparison of the performance of OntoSeg compared to these approaches where number of segments needed (10 segments) is provided[10].

Table 3.4 P_k values for various algorithms in the literature with provided segment number

Approach	3-11	3-5	6-8	9-11
(John et al., 2017)	0.09	0.11	0.005	0.007
U00	0.11	0.13	0.06	0.06
C99	0.13	0.18	0.10	0.10
OntoSeg	0.30	0.19	0.30	0.30
Segmenter	0.36	0.23	0.33	0.43
TextTiling	0.46	0.44	0.43	0.48

The results show that the performance of OntoSeg is not better than most of the state of the art approaches. These results, and the results presented in the previous section conclude that relying on an ontology to semantically represent the text is not sufficient to reveal the meaning behind it and thus, is not practically adequate for the segmentation task. Even with combining the lexical similarity with the ontological similarity, although the performance of OntoSeg was enhanced, it is still not comparative to the state of the art approaches. In order to enhance the understandability of the meaning behind text, there is a need to consider all relations between text blocks.

As argued by (Budanitsky and Hirst, 2006), *relatedness* is more general than *similarity* since dissimilar entities may also be semantically related by other relationships such as meronymy, antonymy, functional relationship or frequent association. Therefore, the performance of OntoSeg needs to be enhanced through improved understandability of text by exploring the semantic relatedness between text blocks rather than using the semantic similarity.

3.5 Chapter Summary

This chapter introduced the OntoSeg algorithm for hierarchical text segmentation. The aim of OntoSeg is to understand the semantic meaning behind text in order to build a conceptual structure out of it. In contrast to traditional text segmentation approaches that rely upon bag-of-words representation of content, OntoSeg uses the semantic interpretation of content to reason about it. OntoSeg uses a Hierarchical Agglomerative Clustering

[10] The results were obtained from (Choi, 2000), (Utiyama and Isahara, 2001) and (John et al., 2017)

(HAC) approach to iteratively cluster text segments that are deemed to be ontologically similar to each other. The output is a tree-like hierarchy of the text. The chapter showed that the produced hierarchy is beneficial in producing hierarchical text segments with different levels of granularity, and also in producing linear text segments by flattening the obtained tree. Experimental results showed that using ontological similarity performs successful segmentation with low error rates. However, comparing its performance against state of the art approaches showed that, although the combination between the lexical and the ontological similarities enhanced the performance of OntoSeg, its performance is not comparative to the state of the art approaches. The results also showed that using lexical similarity only (α = 1 in equation 3.5) is better than using the semantic similarity only in OntoSeg. These results concluded that the performance of OntoSeg needs to be enhanced through improved understandability of text by exploring the semantic relatedness between text blocks rather than using the semantic similarity.

Finally, it is noteworthy to point out that the implementation of the OntoSeg algorithm is publicly available:

- *https://github.com/bayomim/OntoSeg*

and the OntoSeg algorithm along with the experimental work described in this chapter is published in the following paper:

Bayomi, M., Levacher K., Ghorab, M.R., and Lawless, S. "*OntoSeg: A Novel Approach to Text Segmentation using Ontological Similarity*". In the proceedings of the 5th ICDM Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction, ICDM SENTIRE. Held in conjunction with the IEEE International Conference on Data Mining, ICDM 2015. Nov 14th, 2015. Atlantic City, NJ, USA.

4. C-HTS: A Concept-based Hierarchical Text Segmentation approach

Chapter 3 presented OntoSeg, a new approach for hierarchical text segmentation using ontological (semantic) similarity between text constituents. As outlined in the chapter, although OntoSeg was capable of performing successful segmentation with low error rates, it did not produce the best scores compared to the state of the art approaches. Furthermore, as outlined in section 2.3.6, since the work on hierarchical text segmentation is sparse (Wang et al., 2017), there was no publicly available dataset for evaluating hierarchical text segmentation at the time of evaluating OntoSeg. As a result, OntoSeg performance was evaluated on a linear text segmentation dataset. Evaluating a hierarchical text segmentation algorithm using a linear dataset does not give a realistic picture of the performance of the hierarchical algorithm. The reason is that the output of a hierarchical algorithm is a tree structure, while a linear dataset has consequently segmented chunks of text. This means that using consequently segmented text chunks is not adequate to evaluate the quality of the produced hierarchy.

Hence, this chapter proposes *C-HTS*, a new **C**oncept-based **H**ierarchical **T**ext **S**egmentation approach that relies on the semantic *relatedness* between text constituents rather than the semantic *similarity* used in OntoSeg. The core idea of C-HTS is the use of external knowledge to enhance the text representation by adding a semantic layer of concepts that represents the text in a high dimensional semantic space where the *relatedness* between the atomic units of text (text blocks) is measured using this semantic representation. C-HTS relies on the *explicit* semantic representation of text, a method that replaces keyword-based text representation with concept-based features, automatically extracted from massive human knowledge repositories such as Wikipedia. C-HTS represents the meaning of a piece of text as a weighted vector of knowledge concepts, in order to reason about text.

The performance of C-HTS is evaluated on two datasets that are designed specifically for the evaluation of hierarchical text segmentation approaches. Furthermore, the performance of C-HTS is compared against the state of the art hierarchical text segmentation approaches. The results showed that C-HTS performed favourably against these approaches and also outperformed OntoSeg.

Additionally, different lexical similarity measures were used with C-HTS to assess the effectiveness of using the semantic relatedness approach in C-HTS. Experimental results showed that using semantic relatedness outperforms other similarity measures. Furthermore, to asses the effictiveness of using Wikipedia as the underlying knowledge source, an experiment was carried out where WordNet (Miller, 1995) is used as the knowledge source for C-HTS. Experimental results showed that relying on lexical resources such as WordNet offers little information about the different word representations and hence, deteriorates the performance of C-HTS.

Since C-HTS relies on a knowledge space built from Wikipedia, and since Wikipedia is continuously growing, the impact of its growth on segmentation performance is measured. Three different snapshots of Wikipedia from different years are used in order to achieve this. The experimental results show that an increase in the size of the knowledge base leads, on average, to greater improvements in hierarchical text segmentation.

4.1 State of the Art Influences

This section discusses how the limitations of the state of the art approaches influenced the design aspects of C-HTS as a segmentation approach that uses the semantic representation of text.

Semantic representation of text helps in understanding the underlying meaning of its constituents. As already outlined in Chapter 2, most existing text segmentation approaches depend primarily on traditional bag-of-words representations of text in order to build a hierarchical structure (Yaari, 1997; Angheluta et al., 2002; Eisenstein, 2009; Du et al., 2013; Kazantseva & Szpakowicz, 2014). They mainly rely upon the exact matches between words to measure the coherence between two segments in text. Such approaches, however, fail to recognise relevant segments that do not share words with each other. One reason for this is that these approaches treat words in text segments as if they are independent, although it is clear that they are not (Vinokourov et al., 2003). As a result, some approaches began to enrich the text representation by exploiting its semantic meaning. (Choi et al., 2001), for example, enriched their linear segmentation approach, C99 (Choi, 2000), by using Latent Semantic Analysis (LSA) (Deerwester et al., 1990). They applied latent concept modelling to the similarity metric of C99, and showed that using LSA improved the quality of their segmenter. However, LSA-based approaches require a very large corpus, and consequently the pre-processing effort required is significant.

Some other approaches started, on the other hand, to use external resources to enrich text (John et al., 2017). (Stokes et al., 2004) proposed SeLeCT, a news story segmentation approach that uses the WordNet thesaurus (Miller, 1995) as an external lexical resource to add semantic links between words to create lexical chains from these links with respect to a set of chain membership rules. However, the use of such lexical resources offers little information about the different word representations. Furthermore, such resources cover only a small fragment of the language lexicon.

With the advent of the Semantic Web (Berners-Lee et al., 2001), ontologies have been widely used in different tasks to give a conceptual representation of entities (Bayomi & Lawless, 2016). Recently, some approaches have emerged that segment text by exploiting the conceptual representation of its constituent terms. For example, the OntoSeg approach (Bayomi et al., 2015), which was introduced in Chapter 3 of this thesis, relies on the DBpedia ontology to measure the semantic similarity between text blocks. Another approach that relies on ontologies for linear text segmentation was proposed by (Naili et al., 2016). They integrated a domain ontology in the topic segmentation in order to add external semantic knowledge to the segmentation process. They proposed two topic segmenters called TSS-Ont and TSB-Ont based on C99 (Choi, 2000) and TextTiling (Hearst, 1994) algorithms respectively. They used the same techniques as C99 and TextTiling but replaced lexical similarity with concept (semantic) similarity and evaluated their approach against different state of the art apporaches including OntoSeg.

Although these approaches relied on an external resource and used an ontology to add a semantic layer to the segmentation process, they suffer from some drawbacks, such as: they solely extract named entities from text. For a text with few entities or with poor performance from the named entity extraction algorithm, measuring the similarity between text blocks is not feasible. Furthermore, these approaches measure the semantic similarity between entities rather than the semantic relatedness. As argued by (Budanitsky & Hirst, 2006), *relatedness* is more general than *similarity* as dissimilar entities may be semantically related by other relationships such as meronymy (*car–wheel*), antonymy (*hot–cold*), or just by any kind of functional relationship or frequent association (*pencil–paper, penguin–Antarctica, rain–flood*). Another drawback of these approaches is that considering only entities in text does not necessarily reveal much knowledge about the meaning beyond it. (Buchanan & Feigenbaum, 1982) stated that: "*The power of an*

intelligent program to perform its task well depends primarily on the quantity and quality of knowledge it has about that task."

Hence, this chapter proposes C-HTS, a hierarchical model of text segmentation that uses the semantic *relatedness* between text blocks. C-HTS uses the explicit semantic representation of text to measure how text blocks are semantically related based on concepts from a knowledge base. C-HTS uses the exogenous knowledge (externally supplied), rather than the endogenous knowledge extracted from the documents themselves. The approach uses Wikipedia as an external knowledge base to enrich the text representation in a very high-dimensional space of concepts.

4.2 Intuition behind C-HTS

When a person reads a text, the eyes read the words (the lexical representation of text) and send these words to the human's cognitive system, the brain. The brain starts to make sense of these words based on the knowledge of the reader. For example, the name "*Albert Einstein*" in a text document is read by the eyes and then sent to the brain, which starts to map the name to the different concepts that the person knows about Einstein such as: "*Theory of Relativity*", "*Physics*", "*Nobel Prize*", etc. The information that the brain maps the name to, is dependent upon how much knowledge this person has. If the individual does not know about Einstein, the brain would make no sense of that name. The individual could potentially ask other people who have different collections of knowledge for assistance, creating an intellectual representation through collaboration. In this research, the C-HTS algorithm is trying to recreate this methodology in a segmentation algorithm. This research contention that using this approach to understand text is a more effective method for text segmentation and for building a reasonable hierarchical structure from documents.

The essential task in any text segmentation algorithm is to measure the coherence between two adjacent text blocks. Being inherently limited to lexical representation, current approaches cannot reveal much about the coherence between text blocks. Consider the following two sentences for example:

- *Albert Einstein is a German scientist who was born on the 14th of March 1879.*
- *Mileva Marić was born on December 19, 1875 into a wealthy family in Titel, Serbia.*

Lexically, the two sentences are not similar because they mention different names, cities and dates. For a segmentation approach that solely relies upon a lexical representation of

text, the two sentences are not similar to each other. Even for an approach that uses a learning model to learn text representation, if it has not seen the entities mentioned in sentences together in a training set, it will be difficult for it to infer any relation between the two sentences. In fact, Mileva Marić is Einstein's ex-wife, they both worked in physics, she was the only woman among Albert Einstein's fellow students at Zürich's Polytechnic, and they had three children. Hence, an ideal approach to reveal such information about the two sentences, and to measure their relatedness, would use the explicit semantic representation of text based on a knowledge base. Such a knowledge base should be based on human cognition and be intuitive to use and reason over, with no limits on domain coverage or conceptual granularity. Creating and maintaining such a knowledge base requires enormous effort on the part of many people. Luckily, such a collection already exists in the form of Wikipedia, which is one of the largest knowledge repositories on the Web. Hence, relying on such human-organised intensive knowledge reveals more meaning of the text that we want to segment regardless of the approach (linear or hierarchical) or the algorithm that is used for segmentation.

4.3 Semantic Relatedness

The core idea of C-HTS is the use of an external knowledge base to enrich text representations in order to measure the semantic relatedness between terms, and thus sentences, and to utilise this in hierarchical text segmentation. The purpose of measuring semantic relatedness is to allow computers to reason about text. Various approaches have been proposed in the literature to measure the semantic relatedness between terms using an external knowledge source. Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007) is a method that represents meaning in a high-dimensional space of concepts, automatically driven from human-built knowledge repositories such as Wikipedia. ESA defines concepts from Wikipedia articles e.g., ALBERT EINSTEIN and COMPUTER SCIENCE. A target term is essentially represented as a vector of concepts in Wikipedia based on how this term is mentioned in the concept's article. Relatedness is then calculated as the cosine similarity between the two vectors of the target terms (see next section for more details).

Another approach that uses the link structure of Wikipedia to measure semantic relatedness is the Wikipedia Link-based Measure (WLM) (Witten and Milne, 2008). WLM measures the relatedness between two terms using the links found within their corresponding Wikipedia articles rather than using the articles' textual content.

The notion behind using explicit semantic relatedness is that it relies on a knowledge base that is built and continuously maintained by humans. The knowledge base used in this research is Wikipedia, the largest and fastest growing encyclopaedia in existence. This knowledge base is a collaborative effort that combines the knowledge of hundreds of thousands of people. In this research, ESA is used as the approach for measuring the semantic relatedness between text segments. ESA has been widely used in a variety of tasks such as semantic relatedness calculation (Gurevych et al., 2007), concept-based information retrieval (Egozi et al., 2011; Jungwirth & Hanbury, 2018) and text classification (Chang et al., 2008) among other tasks. The efficacy of ESA has been proven compared to other approaches that do not rely on explicit knowledge bases.

4.3.1 How does Explicit Semantic Analysis work?

As mentioned above, ESA relies on a concept space built from a knowledge base, such as Wikipedia, to measure the semantic relatedness between two terms (or text blocks). In Wikipedia-based ESA, a given word is described by a vector which stores the word's association strengths to Wikipedia-derived concepts. A concept is a Wikipedia article (e.g. ALBERT EINSTEIN). This concept is represented as a vector of the terms which occur in that article. Each term, in that vector, is assigned a weight using the *tf-idf* scheme (Salton & McGill, 1986). These weights quantify the strength of association between terms and concepts. After generating terms from the concept article, an inverted index is created that maps each term to a list of concepts in which this term appears. Thus, each word appearing in the Wikipedia corpus can be seen as triggering each of the concepts it points to in the inverted index, with the attached weight representing the degree of association between that word and the concept. The name, Explicit Semantic Analysis, stems from the way vectors are comprised of concepts that are manually defined, as opposed to the mathematically derived contexts used by Latent Semantic Analysis. The processing of Wikipedia articles and building of the concept space is depicted in Figure 4.1. In this example, terms are extracted from the Wikipedia article (Economy). Terms such as: "*market*", "*trade*", "*property*", etc. Each of these terms is indexed in a database and mapped to a list of concepts (articles) in which this term appears along with the *tf-idf* score of the term in that article. For example, one of the concepts that the term "*market*" is mapped to is "*Bazaar*" with 0.72 score. This means that the word "*market*" appears in the "*Bazaar*" article and its *tf-idf* score in that article is 0.72.

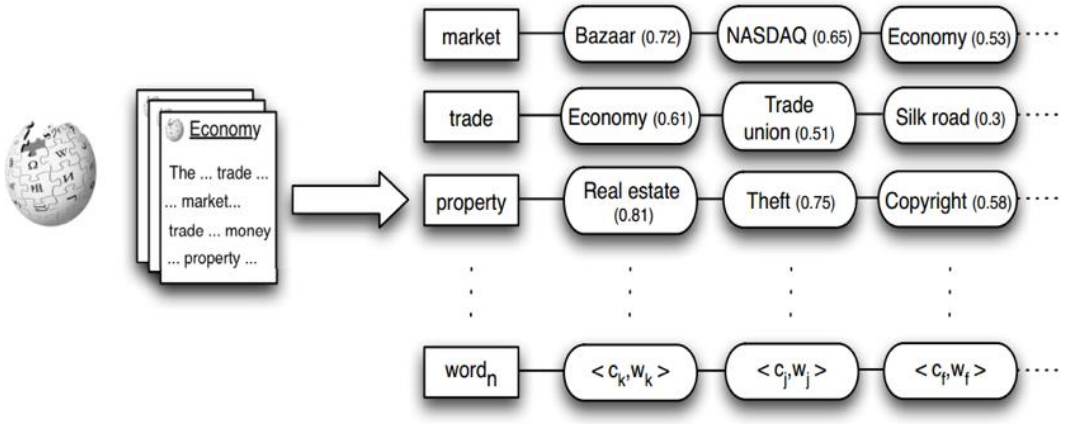


Figure 4.1 The process of generating an ESA model from Wikipedia articles (Egozi et al., 2011).

After building such a concept space, each input term in a text processing task (e.g. segmentation) can be represented as a vector of concepts that the term is associated with, accompanied by the degree of association between the term and each concept. The semantic relatedness between two given terms is measured by computing the cosine similarity between the concept vectors of the two terms. For larger text fragments (a sentence or a paragraph), a concept vector is retrieved for each term in the fragment, then the semantic relatedness between two text fragments is measured by computing the cosine similarity between the centroid of the vectors representing the two fragments. The centroid vector of a text fragment is built based on ranking all the Wikipedia concepts by their relevance to the fragment (Han and Karypis, 2000). Figure 4.2 illustrates the semantic interpretation of two given texts and how the semantic relatedness between their centroid vectors is measured. Given a text fragment (sentence or paragraph), the fragment is represented as a vector using *tf-idf*. For each term in this text fragment, a vector of corresponding entries from the inverted index (the concept space) is retrieved. The retrieved vectors are merged into a weighted vector of concepts that represents the given text. Let S be the set of terms in the input text fragment after removing stop words. Let $\vec{t}$ be the vector of weights for concepts associated with term t in the concept space. The centroid vector $\overrightarrow{C}$ is defined as:

$$\overrightarrow{C} = \frac{1}{|S|} \sum_{t \in S} \vec{t} \qquad 4.1$$

where $|S|$ is the length of vector S that is used for normalisation in order to account for text units of different lengths. The relatedness between two centroid vectors C_i and C_j of two text fragments is computed using the cosine measure:

$$\cos(\overrightarrow{C_i}, \overrightarrow{C_j}) = \frac{\overrightarrow{C_i} \cdot \overrightarrow{C_j}}{\|\overrightarrow{C_i}\| \, \|\overrightarrow{C_j}\|} \quad 4.2$$

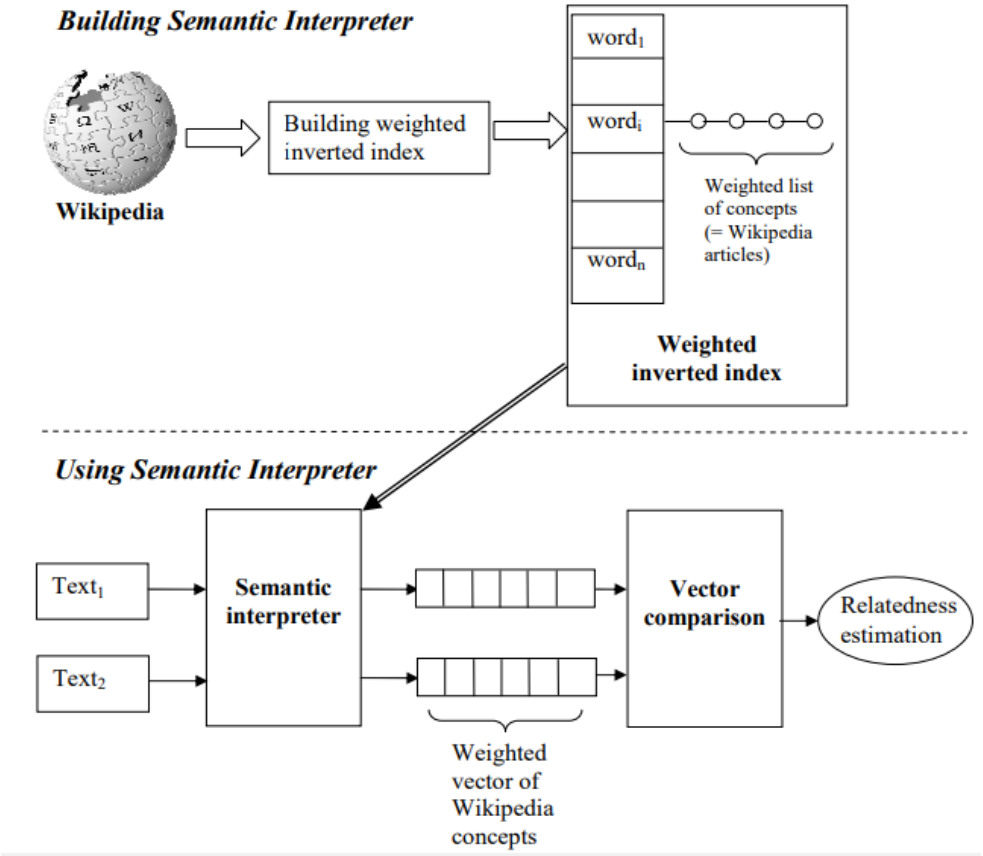


Figure 4.2 Semantic interpretation of two text units using ESA (Gabrilovich and Markovitch, 2007)

To elaborate on the notion of semantic relatedness using ESA, consider the two sentences in the example mentioned earlier in section 4.2:

- *Albert Einstein is a German scientist who was born on the 14th of March 1879.*
- *Mileva Marić was born on December 19, 1875 into a wealthy family in Titel, Serbia.*

After applying morphological analyses (see section 4.4.1) on the two sentences, each remaining term in each sentence is mapped to a vector of concepts from the vector space. Each sentence is then represented as the centroid of the vectors of the sentence's terms (Han & Karypis, 2000). For the first sentence, the centroid of the vectors contains the following concepts (among other concepts):

- ALBERT EINSTEIN AWARD
- THE EVOLUTION OF PHYSICS
- HANS ALBERT EINSTEIN → (*second child and first son of Albert Einstein and Mileva Marić*)
- ELSA EINSTEIN → (*the second wife of Einstein*)

And the centroid of the vectors of the second sentence contains the following concepts (among other concepts):

- MILEVA MARIĆ
- HANS ALBERT EINSTEIN
- ELSA EINSTEIN
- EINSTEIN FAMILY

From these vectors, we can see that the concept vectors of the two sentences have concepts in common and measuring the cosine similarity between them (Equation 4.2) can show that although the two sentences are not lexically similar, they are semantically related to each other.

4.4 C-HTS Architecture

The architecture of C-HTS consists of three phases:

1- Morphological Analysis

2- Semantic Representation and Relatedness Measuring

3- Hierarchical Agglomerative Clustering

4.4.1 Morphological Analysis

In this phase, the target text is processed to be split into sentences and to have stopwords[1] removed, as they are generally assumed to be of less, or no, informational value. The remaining words are then stemmed and converted into their root using the Porter stemmer (Porter, 1980). This morphological analysis technique has been used in processing the Wikipedia terms and concepts while building the concept space from Wikipedia (section 4.3.1). The remaining terms are then used as input for the next phase.

[1] Since the research in this thesis is not focused on a specific domain, a general domain stopwords list for English is used with C-HTS.

4.4.2 Semantic Representation and Relatedness Measuring

The key idea in C-HTS consists of treating the segmentation of text as an examination of the semantic relatedness between text blocks rather than traditional lexical similarity. A text block is the elementary unit of the segmentation algorithm, which is one sentence in C-HTS. For each sentence, and for each term in that sentence, the term is mapped to a vector of concepts from the concept space that was created from Wikipedia. The semantic relatedness between two (adjacent) sentences is calculated as the cosine similarity between the centroid of the vectors representing the individual terms in each sentence using Equation 4.2.

4.4.3 Hierarchical Agglomerative Clustering

As in OntoSeg (Chapter 3), C-HTS transfers the agglomerative clustering technique from document level to text level (section 3.2.3). The difference between the two approaches is that while OntoSeg measures coherency between two adjacent clusters (text blocks) based on their semantic similarity, C-HTS measures the coherency based on their semantic relatedness. The main topic for research in HAC algorithms is the proximity test. In C-HTS, the semantic relatedness between text blocks is applied as the proximity test. By applying hierarchical agglomerative clustering on text blocks the algorithm successively agglomerates blocks that are deemed to be semantically related to each other, thus forming a text structure. C-HTS uses HAC because it is a bottom-up clustering approach. The idea behind using a bottom-up approach in text segmentation is that it starts from the smallest clusters (sentences), that are considered the seeds of the text, and then builds the text structure by successively merging the semantically coherent clusters. This way of building the document structure can be regarded as a hierarchically coherent tree that is useful to support a variety of content discoverability methods as it provides different levels of granularity for the underlying content.

In the dendrogram depicted in Figure 4.3, we can see that in each iteration of C-HTS a new level (horizontal dotted lines) is constructed from the agglomeration process on the previous level. Each level is considered a different representation of the document granularity. The level of granularity increases as we move from the root to the bottom of the tree (the leaves). For example, in level 5 in the dendrogram, we can see that the document at that level of granularity can be segmented into two segments with boundaries 19 & 25, i.e. the document can be segmented at sentence 19 and sentence 25. Hence, for

an adaptive system that requires a specific number of sentences based on its user needs (such as reading speed), traversing such a tree would allow us to easily extract such segment based on the relevance of the segment to the adaptive system's query and the required length. This segment was constructed not based on the HTML or paragraphs structure of the document, rather, it was constructed based on the semantic relation between its constituents.

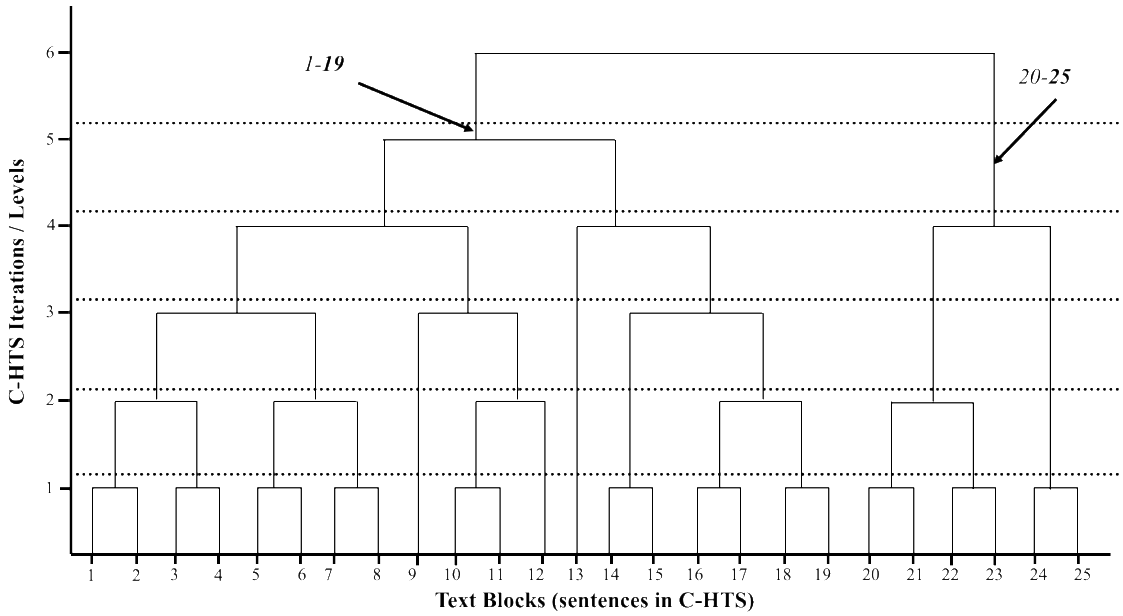


Figure 4.3 C-HTS output as a dendrogram of a sample text

4.4.4 Word Sense Disambiguation

Word Sense Disambiguation (WSD) (section 2.2.3) is the task of identifying the meaning of a term, when the term has multiple meanings, based upon the context of where it appears (Navigli, 2009). For example, "light" can mean "not heavy" or "illumination", what identifies its meaning is the context of where "light" is used. For a natural language processing task like text segmentation, disambiguating such words would allow the task to better understand the meaning of the sentence and to reason about it and thus enhance the quality of the segmentation. For lexical segmenters, being inherently limited to lexical representation of text, these approaches require an extra level of sophistication to disambiguate words.

In C-HTS, the relatedness between sentences is measured as the cosine similarity between the centroid of the vectors representing the two sentences. This representation of text is considered an implicit disambiguation of terms. For example, consider a sentence that has the term "*Apple*" amongst other computer related terms. Taking the centroid of the

vectors will boost the computer-related concepts and will disambiguate the term effectively.

To illustrate how words are disambiguated in C-HTS, consider the following sentence:

- "*I love fruit, particularly a nice apple*".

In this sentence, after applying morphological analysis (section 4.4.1), the remaining prominent terms are *love*, *fruit*, *particularli*, *nice* and *appl*[2]. Among these terms, the term "*appl*" (apple after stemming) has different interpretations. From the underlying concept space that have been created from Wikipedia, the top concepts generated for the term *appl* (among other concepts) are:

- APPLE DAY → (*related to apple fruit*)
- APPLE SPECIALIST → (*related to Apple Inc.*)
- APPLE EXTENDED KEYBOARD → (*related to Apple Inc.*)
- EMPIRE (APPLE) → (*related to apple fruit*)
- APPLE STORE (ONLINE) → (*related to Apple Inc*.)

The majority of the top concepts are related to the company, Apple Inc. However, when considering the centroid of the vectors representing the whole sentence, the top generated concepts are (among other concepts):

- FRUIT PICKING
- ROME APPLE → (*a kind of apple originating near Rome Township, Ohio*)
- LIST OF APPLE CULTIVARS
- EMPIRE (APPLE) → (*a kind of apple derived from a seed grown in 1945*)

From these concepts, we can see that they all are related to the fruit apple. This illustrates that considering the centroid of the vectors of a sentence disambiguates the terms without adding extra sophisticated text processing layers. This vector can also be seen as a representation of the context of that sentence. This in turn enhances the understandability of text and enhances the segmentation quality.

[2] These terms are the stemmed version of the original ones. For example, the word *Apple* is stemmed to *appl* using Porter stemmer

4.5 Evaluation

Research on hierarchical text segmentation has been scarce and most state of the art research has evaluated their hierarchical approaches on linear text segmentation datasets. For example, (Yaari, 1997) evaluated his approach on the *Stargazers* article. He compared his approach against a linear text segmentation approach, TextTiling. OntoSeg (Bayomi et al., 2015) (Chapter 3) was evaluated using the linear text segmentation dataset proposed by (Choi, 2000). This is due to the lack of available datasets which are suitable for hierarchical text segmentation evaluation. Evaluating a hierarchical text segmentation algorithm using a linear dataset does not give a realistic picture of the performance of the hierarchical algorithm. The reason is that the output of a hierarchical algorithm is a tree structure, while a linear dataset has consequently segmented chunks of text. Hence, selecting an appropriate dataset is a critical step in the evaluation process.

4.5.1 Datasets

In this thesis, we argue that C-HTS is applying hierarchical text segmentation as if a human would perform the task (section 4.2). To prove this assumption, a gold standard dataset that is created by humans is needed. Furthermore, the dataset needs to be suitable for a hierarchical text segmentation task. Luckily, (Kazantseva and Szpakowicz, 2014) proposed two datasets that are suitable for evaluating hierarchical text segmentation and both were annotated by humans. The authors evaluated their approach, Hierarchical Affinity Propagation for Segmentation (HAPS), against two well-defined datasets: the *Moonstone* dataset and the *Wikipedia* dataset compiled by (Carroll, 2010).

1- ***Moonstone dataset***: This dataset consists of nine chapters of the *Moonstone* novel. (Kazantseva and Szpakowicz, 2014) employed human annotators to annotate the dataset and to identify the hierarchical structure of each text document (in this case, each chapter). The annotators were asked to read a chapter and split it into top-level segments according to where they can see a shift in topic. Each chapter was annotated by 3-6 people (4.8 on average).
2- ***Wikipedia dataset***: This dataset was compiled by (Carroll, 2010). The dataset consists of 66 Wikipedia articles on various topics. The HTML pages were converted to flat text, and unneeded content such as navigation boxes, and image captions were removed. The hierarchical structure for each article is created automatically from the structure of the Wikipedia page, i.e. heading text was replaced with a

boundary marker, indicating the heading depth. This depth represents the level in the text's hierarchical structure. While the levels in the Wikipedia dataset were created using automatic techniques (e.g. Wikipedia *VisualEditor*[3]), the original structure of the documents is identified by the human authors who contribute to Wikipedia[4]. Thus, it is considered a human annotated dataset.

Since C-HTS is based on using an external knowledge base to enrich text representation, evaluating its performance using these two datasets will give us a realistic picture of the performance of C-HTS as a concept-based approach. This is due to the inherent human involvement in the construction process of the two datasets.

4.5.2 Baselines

To evaluate the quality of the hierarchical structure produced by C-HTS, there is a need to compare its performance against hierarchical text segmentation approaches. As mentioned before, work on hierarchical text segmentation has been scarce. To the best of the authors' knowledge, and at the time of evaluating C-HTS, the only publicly available hierarchical segmenter (along with a dataset) is HAPS that was proposed by (Kazantseva and Szpakowicz, 2014). HAPS is a hierarchical text segmentation approach that is based on a graphical model for hierarchical clustering called Hierarchical Affinity Propagation (Givoni et al., 2011). The input for HAPS is a matrix of similarity between text blocks. HAPS requires the desired number of levels to be in the produced topical tree and a preference value for each data point and each level. HAPS also finds a centre for each segment at every level of the produced topical tree, a data point (a sentence) which best describes the segment.

HAPS was compared against two linear segmenters where a hierarchical segmentation was obtained from each approach. The two approaches are *MCSeg* (Minimum Cut Segmenter) (Malioutov and Barzilay, 2006) and *BSeg* (Bayesian based Segmenter) (Eisenstein, 2009). These two systems were chosen because they are representative of the existing text segmentation methods, and their implementations are freely available on the internet. *MCSeg* casts text segmentation in a graph-theoretic framework. In this approach, text is abstracted into a weighted undirected graph, where the nodes of the graph correspond to text blocks and edge weights represent the pairwise block similarity. Text segmentation in *MCSeg* corresponds to a graph partitioning that optimises the

[3] https://en.wikipedia.org/wiki/Wikipedia:VisualEditor [Accessed: April 08, 2018]
[4] https://en.wikipedia.org/wiki/Wikipedia:Who_writes_Wikipedia [Accessed: April 08, 2018]

normalised-cut criterion. In *BSeg*, the lexical cohesion between segments is placed in a Bayesian context. The words are modelled in each topic segment as drawn from a multinomial language model associated with the segment.

To obtain hierarchical segmentation from these two linear segmentation systems, both systems were run first to produce top-level segmentations (sequential segments). Each segment thus computed was a new input document for segmentation. The procedure was repeated twice to obtain a three level structure of the text. In addition to these baselines, we also use OntoSeg as another baseline.

In this research, C-HTS is compared to these three systems as baselines (HAPS, MSSeg and BSeg). For evaluation consistency, we use their experimental settings by evaluating the top three levels (excluding the root) of the document structure produced by each system. Furthermore, *windowDiff* (section 3.4.3) is used as the evaluation metric. windowDiff is designed to evaluate linear text segmentation not hierarchical trees. Hence, in this experiment, and for the sake of comparability we follow the same technique as (Kazantseva and Szpakowicz, 2014). Each level of the text hierarchy is treated as a separate segmentation and each hypothetical level is compared against a corresponding level in the reference segmentation.

4.5.3 Results

The *Moonstone* dataset has on average 4.8 annotations per chapter. To obtain a realistic picture of the results across the different annotators per file, each hypothetical segmentation is separately compared against each available gold standard. After that, the averages across all annotators are taken as the final score. For the two datasets (*Moonstone* and *Wikipeida*), Table 4.1 shows the results of the comparison between C-HTS and the other four baselines using the *windowDiff* evaluation metric. Since C-HTS, OntoSeg and HAPS are inherent hierarchical text segmentation approaches, they were run without knowing the number of segments in each level. Neverthless, HAPS requires the number of levels to be known in advance, which was set to three. *BSeg* was able to run with and without knowing the number of segments. In the results, the *BSeg* run without this parameter is reported. *MCSeg*, on the other hand, required that the exact number of segments be specified. This makes it considerably more informed than others.

The results show that C-HTS performs well on both datasets compared to the baselines, even when compared to more informed baseline (*MCSeg*). For the *Wikipedia* dataset, C-

HTS performs better than the baselines on all three levels. This proves that using the explicit semantic representation of text gives more understanding of the meaning of the text, and thus enhances the process of hierarchical text segmentation. This also proves that using semantic relatedness between text constituents produces better segmentation than using the semantic (ontological) similarity as in OntoSeg.

For the *Moonstone* dataset, C-HTS performs favourably on the top and bottom levels but it is noticed that its performance on the middle level is not better than HAPS and BSeg. We argue that this is because in the *Moonstone* dataset the boundary for each level, in each document, was placed by very few number of annotators (on average 4.8 annotations per chapter), hence, there can be mixed agreement between those annotators on the correct placement of the level boundary. On the other hand, in *Wikipedia* dataset, the original article hierarchy (where levels are obtained from) was created and updated with the agreement of the Wikipedia article contributors.

Table 4.1 Evaluation of C-HTS, HAPS, OntoSeg and iterative versions of MCSeg and BSeg using *windowDiff* per level

	Level	*Moonstone*	*Wikipedia*
C-HTS	3 (top)	**0.320**	**0.330**
	2 (middle)	0.507	**0.397**
	1 (bottom)	**0.488**	**0.402**
HAPS	3 (top)	0.337	0.421
	2 (middle)	**0.422**	0.447
	1 (bottom)	0.556	0.617
OntoSeg	3 (top)	0.366	0.350
	2 (middle)	0.523	0.401
	1 (bottom)	0.544	0.411
MCSeg	3 (top)	0.375	0.440
	2 (middle)	0.541	0.424
	1 (bottom)	0.601	0.471
BSeg	3 (top)	0.600	0.637
	2 (middle)	0.447	0.877
	1 (bottom)	0.545	0.952

4.6 Discussion

4.6.1 Elementary Units for C-HTS

Bottom-up hierarchical text segmentation algorithms start with atomic text pieces as their elementary units and then successively grow areas of coherence. The elementary units can be of a fixed size, such as a specific number of sentences, or can be of a mutable size such as paragraphs. For example, (Yaari, 1997) and HAPS (Kazantseva & Szpakowicz,

2014) used paragraphs as the elementary units for their segmenters, while in C-HTS we use one sentence as the elementary unit.

The size of the elementary units is an influential parameter for the segmentation algorithm and it has implications on the segmentation accuracy. Previously, in Chapter 3 (section 3.4.2), we experimented with the influence of different elementary unit sizes on the hierarchical segmentation task. We experimented with sizes ranging from one to four sentences per unit. The best run reported in the experiments was when we used one sentence as the elementary unit. The results also concluded that the higher the size of the elementary unit, the lower the accuracy of the segmentation[5].

This also adds to the understanding of the inconsistency of C-HTS performance on the *Moonstone* dataset. Besides the disagreement between the few number of human annotators about the correct placement of level boundary, the elementary units presented to the annotators, to build the gold standard, were paragraphs. As a result, and for evaluation consistency, we had to set the elementary units for C-HITS to be paragraphs which impacted the performance of the algorithm. This can be seen in the results of the first experiment (and the following experiments) where the performance of C-HTS on the Wikipedia dataset, where we use one sentence as the elementary unit, gives, on average, lower error rates than its performance on the *Moonstone* dataset.

4.6.2 Text Granularity

Hierarchical text segmentation approaches produce a structural representation of text that represents different levels of granularity. In HAPS, the desired number of levels needs to be passed as a parameter to the algorithm. In contrast, in C-HTS, it does not need to know number of levels that are needed in the output structure because the structure produced by C-HTS depends on the coherence between the atomic units of the text. This way of building the structure makes the output more granular and facilitates its use in different tasks like content discoverability and reusability. Identifying the number of levels of the output limits the usage of the produced hierarchy, as each adaptive system requires a different level of content granularity. Hence, from this point of view, HAPS is considered a system-dependent approach, as its parameters need to be set depending on the system in question. On the other hand, C-HTS is considered a system-independent approach as

[5] We also tried different sizes for the elementary units for C-HTS and the results aligned with the results of OntoSeg, the higher the size of the elementary unit, the lower the accuracy of the segmentation

it produces all the available levels of granularity in the processed document, hence it can produce content that is amenable for reuse in different systems.

4.6.3 Multilingual C-HTS

C-HTS is based on the concept space built from Wikipedia. Wikipedia is the largest encyclopaedia in existence that is available in dozens of languages. As of April 2018, there are 298 Wikipedias of which 288 are active and 10 are not[6]. Building a concept space for these languages would help an ESA-based task to be used with texts in different languages. (Gurevych et al., 2007) applied ESA to the German-language Wikipedia[7] and used it for semantic relatedness and information retrieval tasks. Their experiments showed that using ESA was superior compared to a system based on the German version of WordNet, GermaNet (Hamp and Feldweg, 1997).

The core of C-HTS is the process of measuring the semantic relatedness between clusters using the explicit semantic interpretation of text. This process is essentially based on the underlying concept space that has been built from Wikipedia. Moving C-HTS from one language to another can be done easily. Changing the language of the underlying concept space would make no difference in the running of C-HTS. The only step which must be changed is the morphological analysis (section 4.4.1) to filter out and stem the prominent terms in text. This step is relatively easy to implement as there has been a large volume of work completed on morphological analysis for languages other than English (Manning et al., 2014). Hence, C-HTS can be seen as a multilingual hierarchical text segmentation approach that can semantically represent text and reason about it regardless the language of the text.

Although C-HTS can be applied on multilingual documents to produce a hierarchical structure, the research in this thesis focuses solely upon content resources available in English as a first step and reserves multilingual content processing for future work (section 7.3.1).

4.7 C-HTS Validation

The key idea proposed in C-HTS is to perform the segmentation of text based on the semantic relatedness between its blocks. As discussed in section 4.3, C-HTS uses explicit

[6] https://en.wikipedia.org/wiki/List_of_Wikipedias [Accessed: April 08, 2018]
[7] https://en.wikipedia.org/wiki/German_Wikipedia [Accessed: April 08, 2018]

semantic analysis (ESA) to measure the semantic *relatedness* between text blocks using Wikipedia as its knowledge base. In this research, the experimantal results reported in section 4.5.3 demonstrated the efficiancy of C-HTS in building a hierarchical structure out of textual documents and its competitive performance against the state of the art approaches.

To validate the efficacy of using Wikipedia as the underlying knowledge base for conceptual representation of text in C-HTS, an experiment was carried out where the WordNet thesaurus (Miller, 1995) is used as the underlying knowledge base to add semantic representation of text (phase 2 in C-HTS, section 4.4.2).

Additionally, to validate the efficacy of using the explicit semantic representation of text rather than its lexical representation, another experiment was carried out where the lexical similarity, in contrast to semantic relatedness, between text constituents is measured in C-HTS.

4.7.1 Semantic Similarity using WordNet

WordNet[8] (Miller, 1995) is a broad coverage lexical network of English words. Nouns, verbs, adjectives, and adverbs are each organised into networks of synonym sets (called *synsets*) that each represent one underlying lexical concept and are interlinked with a variety of relations (Budanitsky and Hirst, 2006). Over time, different versions of WordNet have been proposed that cover languages other than English, such as EuroWordNet[9] (Vossen, 1998) which covers several European languages (Italian, Spanish, etc.) and GermaNet[10] (Hamp and Feldweg, 1997) which covers the German Language. Different NLP approaches relied on WordNet as their source for semantic representation of text (Stokes, Carthy and Smeaton, 2004; Lu et al., 2015). However, as discussed earlier (section 4.1), the use of lexical resources (e.g. WordNet) offers limited information about the different word representations. Furthermore, such resources cover only a small fragment of the language lexicon.

To assess this assumption, an experiment was carried out where WordNet is used as the underlying knowledge base for C-HTS. Additionally, different concept similarity metrics are used in this experiment:

[8] https://wordnet.princeton.edu/ [Accessed: March 28, 2018]
[9] http://projects.illc.uva.nl/EuroWordNet/ [Accessed: April 08, 2018]
[10] http://www.sfs.uni-tuebingen.de/GermaNet/ [Accessed: April 08, 2018]

1- *Path similarity* (Rada et al., 1989): computes shortest number of edges from one word sense to another in WordNet hierarchical structure. Using edge counting (section 3.2.2.1), the distance between two disjunctive sets of concepts is defined as the minimum path length from any element of the first set to any element of the second.
2- *Leacock-Chodorow Similarity (LCH)* (Leacock and Chodorow, 1998): the same as the Path similarity except that it uses the negative logarithm of the result of Path similarity.
3- *Wu-Palmer similarity* (WUP) (Wu and Palmer, 1994): similar to LCH, except it weights the edges based on distance in the hierarchy (section 3.2.2.1).
4- *The Lesk similarity* (Lesk, 1986): it defines the similarity between two concepts as a function of the overlap between the corresponding definitions, as provided by a dictionary such as WordNet.

The WS4J Library[11] (Shima, 2014) is used in this experiment.

Table 4.2 shows that the performance of C-HTS using Wikipedia as its knowledge base outperforms its performance using WordNet even with different relatedness measures used with WordNet. This proves that using Wikipedia as a large knowledge base that is built from the collaborative work of hundreds of thousands of people is better than relying on a limited knowledge base such as WordNet.

Table 4.2 Comparison between different similarity measures using WordNet in C-HTS

	Level	*Moonstone*	*Wikipedia*
Wikipedia $_{ESA}$	3 (top) 2 (middle) 1 (bottom)	**0.320** **0.507** **0.488**	**0.330** **0.397** **0.402**
WordNet $_{Path}$	3 (top) 2 (middle) 1 (bottom)	0.393 0.523 0.523	0.385 0.412 0.421
WordNet $_{LCH}$	3 (top) 2 (middle) 1 (bottom)	0.393 0.525 0.520	0.385 0.410 0.422
WordNet $_{WUP}$	3 (top) 2 (middle) 1 (bottom)	0.397 0.523 0.522	0.378 0.412 0.424
WordNet $_{Lesk}$	3 (top) 2 (middle) 1 (bottom)	0.375 0.508 0.536	0.377 0.411 0.420

[11] https://github.com/Sciss/ws4j/ [Accessed: January 22, 2018]

4.7.2 Lexical Representation

Lexical representation has been widely used in the literature in text segmentation (Hearst, 1994; Choi, 2000). As its name suggests, it splits text into segments based on words that these segments share with each other. Lexical cohesion refers to the connectivity between two portions of text in terms of word relationships. It relies mainly upon the endogenous knowledge extracted from the documents themselves. Text segmentation approaches that rely upon lexical similarity between text blocks, however, fail to recognise relevant segments that do not share words with each other. Hence, in C-HTS, the semantic relatedness between text blocks is employed to reveal much knowledge about the meaning beyond text.

In order to assess the efficacy of using the semantic representation of text in C-HTS, an experiment was carried out where the lexical representation of text is used to measure the lexical similarity rather than using the semantic relatedness between text blocks (second phase in C-HTS, section 4.4.2). Additionally, different lexical similarity measures are used in this experiment:

1- *Cosine Similarity* (Singhal, 2001): a basic measure often used in information retrieval, weights words according to their term frequencies scores, and computes the cosine between two text vectors.
2- A string distance metric such as *Levenshtein distance* (Levenshtein, 1966): it measures the similarity between two given strings based on the distance between them. The distance is the number of deletions, insertions, or substitutions required to transform the first string into the second.
3- *Monge-Elkan measure* (Monge and Elkan, 1996): is a simple but effective method for measuring the similarity between two strings containing multiple tokens, using an internal similarity between tokens. It measures the average of the similarity values between pairs of more similar tokens within two given strings.
4- *Longest Common Subsequence (LCS)* (Allison and Dix, 1986): refers to the longest string two texts have in common, when gaps between the series in characters are allowed.

The Dkpro Similarity Framework[12] (Bär et al., 2013) is used in this experiment.

[12] https://dkpro.github.io/dkpro-similarity/ [Accessed: January 22, 2018]

Table 4.3 shows that the explicit semantic representation of text (ESA) outperforms the lexical representation approach in all similarity measures, segmentation levels and in both datasets. This in fact is not surprising as lexical representation approaches can process only the information that they can 'see'. While, on the other hand, explicit semantic representation of text allows a NLP task (e.g. segmentation) to reason about text using knowledge extracted from a massive knowledge base such as Wikipeida.

Table 4.3 Comparison between different coherency measures used with C-HTS

	Level	*Moonstone*	*Wikipedia*
ESA	3 (top)	**0.320**	**0.330**
	2 (middle)	**0.507**	**0.397**
	1 (bottom)	**0.488**	**0.402**
Cosine	3 (top)	0.352	0.362
	2 (middle)	0.510	0.401
	1 (bottom)	0.499	0.407
Monge-Elkan	3 (top)	0.406	0.430
	2 (middle)	0.528	0.410
	1 (bottom)	0.517	0.418
LCS	3 (top)	0.399	0.427
	2 (middle)	0.541	0.420
	1 (bottom)	0.522	0.429
Levenshtein	3 (top)	0.428	0.451
	2 (middle)	0.523	0.420
	1 (bottom)	0.537	0.431

4.8 The Impact of Knowledge Breadth

In this research, C-HTS uses a concept space that is built from the text of a knowledge base articles (Wikipedia). (Anderka and Stein, 2009) showed that the size of the text collection used to build the concept space has much more impact on the explicit semantic analysis performance than its nature (how its written or organised). Wikipedia is being constantly expanded and updated by different contributors who add new articles and extend the existing ones. Consequently, the amount of knowledge in Wikipedia is expanding. We conjecture that such expansion, and the growth of information available in the knowledge base should impact the accuracy of the segmentation process. To test this assumption, different snapshots of the entire Wikipedia knowledge base were acquired from three different years: 2006, 2013 and 2017. The snapshots from 2006 and 2013 were processed by (Carvalho et al., 2014) and ready for use. For the 2017 snapshot, we processed

it ourselves[13] following the instructions in (Gabrilovich and Markovitch, 2009) and (Carvalho et al., 2014)[14].

4.8.1 Experiment and Results

Table 4.4 presents a comparison of the amount of information contained in the three used Wikipedia snapshots. In this experiment, C-HTS was run on the two aforementioned datasets but using different concept spaces built from the three different Wikipedia snapshots. The purpose of this experiment is to examine the effect of the expansion of the underlying knowledge base has on C-HTS.

Table 4.5 shows the results of the experiment. As can be observed, increasing the amount of knowledge in the knowledge base leads, on average, to improvements in hierarchical text segmentation. Although the difference in performance of the three versions is admittedly small, it is consistent across the datasets.

Table 4.4 Comparison of the three Wikipedia snapshots

Snapshot's Year	2006	2013	2017
# Articles	895,000	4,133,000	5,373,241
# Concepts used	369,767	1,270,521	1,446,243
# Distinct terms	598,391	1,615,525	1,825,353
Concept space size	11 Gb	21 Gb	12.5 Gb[15]

Table 4.5 Comparison of the three Wikipedia snapshots

	Level	*Moonstone*	*Wikipedia*
2006 Snapshot	3 (top)	0.347	0.365
	2 (middle)	0.545	0.404
	1 (bottom)	0.504	0.411
	Average	0.465	0.3933
2013 Snapshot	3 (top)	0.346	0.366
	2 (middle)	0.539	**0.397**
	1 (bottom)	0.509	0.405
	Average	0.464	0.390
2017 Snapshot	3 (top)	**0.320**	**0.330**
	2 (middle)	**0.507**	**0.397**
	1 (bottom)	**0.488**	**0.402**
	Average	**0.438**	**0.3823**

[13] Wikipedia 2017 snapshot processed for ESA is available here: https://goo.gl/JZhEvm
[14] The technical instructions and snapshots can be found here: https://github.com/dscarvalho/easyesa/ [Accessed: December 3, 2018]
[15] We indexed the 2017 snapshot in MongoDB v3 that uses the WiredTiger storage engine which applies more compression than the old *mmapv1* engine in MongoDB version used in indexing both 2006 and 2013 snapshots.

4.9 Chapter Summary

This chapter introduced C-HTS, a new Concept-based Hierarchical Text Segmentation approach. The core idea of C-HTS is the use of an external knowledge to enhance the text representation by adding a semantic layer of concepts that represents the text in a high dimensional semantic space. Relatedness (in contrast to similarity) between the atomic units of text is measured using this semantic representation. A Hierarchical Agglomerative Clustering (HAC) algorithm is then used to grow coherent segments of the text. The output of C-HTS is a tree-like structure of the input text. C-HTS was compared against a set of state of the art approaches across two different datasets. The results showed that C-HTS performed favourably against other approaches. Additionally, to asses the effictiveness of using Wikipedia as the underlying knowledge source, an experiment was carried out where WordNet is used as the knowledge source for C-HTS. Experimental results showed that relying on resources such as WordNet offers little information about the different word representations and hence, deteriorates the performance of C-HTS. Furthermore, different lexical similarity measures were used with C-HTS to assess the effectiveness of using the semantic representation of text in C-HTS. Experimental results showed that using semantic relatedness outperforms other similarity measures.

Another experimrnt was carried out to evaluate the influence of the size of the knowledge base that C-HTS uses to reason about text. Since C-HTS uses Wikipedia as the underlying knowledge base, its performance was measured when using different concept spaces built from different snapshots of Wikipedia over different years: 2006, 2013 and 2017. The results showed that there is a measurable impact upon segmentation performance, and while the difference is small, it is consistent across the two datasets.

The concept space that was built from the 2017 Wikipedia snapshot is publicly available:

- https://goo.gl/JZhEvm

and the implementation of the C-HTS algorithm is also publicly available:

- https://github.com/bayomim/C-HTS

Finally, it is noteworthy to point out the C-HTS algorithm along with the experimental work[16] described in this chapter was published in the following paper:

[16] Except for experiments 2 and 3 in section 4.7

Bayomi, M. & Lawless, S. "*C-HTS: A Concept-based Hierarchical Text Segmentation approach*". In the Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). Miyazaki, Japan: European Language Resources Association (ELRA).

5. CROCC: Customised Reuse of Open- and Closed-corpus Content

The previous two chapters presented different hierarchical text segmentation approaches that use semantic text representation to build a structure out of unstructured textual content. Evidence from the experiments discussed in the previous chapters showed that C-HTS (Chapter 4) outperformed OntoSeg (Chapter 3) in the task of building a hierarchical structure of text (section 4.5.3).

The second research question posed by this thesis is:

Can the produced structure be used to enhance the discoverability and reusability of content for adaptive systems?

In order to study the benefit of the produced structure in a content adaptation scenario, this chapter describes a novel content-supply service named *CROCC* (**C**ustomised **R**euse of **O**pen- and **C**losed-corpus **C**ontent). CROCC is a service which harvests content resources from open and closed corpora[1] in their native form, builds a structure out of each content resource based on its conceptual representation, and delivers content slices[2] which meet the requirements of individual adaptive systems. This service is focused upon contributing the third objective of this thesis (RO 3) (section 1.2).

5.1 State of the Art Influences

The analysis conducted in the state of the art chapter (Chapter 2), influenced various design aspects of the content-supply service which is described in this thesis. The aim of this section is to present a summary of these influences and how they affect the core properties of the content-supply service developed by this research.

Influences derived from the state of the art review are presented and grouped within seven different categories, each highlighting individual perspectives with respect to the limitations and opportunities identified as part of the analysis performed. These categories are:

1. Content Incorporation
2. Content Right-Fitting
3. Content Structuring

[1] This thesis recognises that there are key challenges regarding IP and copyright when attempting to reuse open corpus content, even for educational use. Initiatives such as creative commons can potentially have a very positive impact on clarifying the copyright permissions for educational use. However, in this thesis, copyright and IP issues are deemed to be out-of-scope.

[2] A slice is a piece of content (one or more sentences) that originates from a pre-existing content resource.

4. Content Representation
5. Content Indexing
6. Content Discoverability
7. Content Reusability

These categories are detailed in the following subsections.

5.1.1 Content Incorporation Techniques

- Many adaptive systems have mainly relied upon the manual incorporation of content resources (section 2.4.4.1).
- However, these systems have failed to overcome the problems associated with extending and updating the content they can provide to their users. Hence, they do not scale well and are deemed impractical for a variety of real-world applications.
- Extending resources in such systems is a manual and labour-intensive task that needs much time and effort from a domain expert.
- This in turn makes content very tightly coupled to these systems and, as a result, strongly impedes the reusability of this content in different systems.
- Additionally, whenever a change in the structure or presentation of content resources needs to be made, adaptation techniques and algorithms developed within the adaptive system must be altered accordingly or replaced altogether, which can be quite labour intensive.

Key Principle 1.a: Content should be **incorporated automatically** and should be made available to adaptive systems through a service that can deliver resources as content packages[3] that are amenable for reuse within various independent adaptive systems.

Key Principle 1.b: The service should be designed using a **flexible architecture** that allows plugging-in, removing, enabling, or disabling alternative components or algorithms based on the content being processed by the service.

5.1.2 Content Right-Fitting

- Adaptive systems started to employ web technologies to automatically retrieve and incorporate content from the sources reachable through the web and deliver them to their users (section 2.4.4.2).

[3] Content package is a piece of content extracted from a document, as opposed to a whole document in its entirety.

- While web resources are often freely available, they are generally not properly structured into easy-to-handle units. The one-size-fits-all nature of web content makes it "same content for all people".
- For this reason, the incorporation and delivery of open corpus content resources in their native form is inadequate for adaptive systems (section 2.5.2.1).

Key Principle 2: Resources should be **structured into fine grained content packages** to support the creation of content slices to meet the adaptation requirements of individual users or applications. This would give much more control to the adaptive system over the provided content.

5.1.3 Content Structuring

- Several approaches started to overcome the limitations associated with the one-size-fits-all nature of content resources by focusing upon extracting content fragments from harvested resources (section 2.5.2.2).
- They have typically relied upon the original structure of the content resources (e.g. the HTML structure of web pages). This means that these approaches do not consider the content itself, rather, they consider the structure with which the content is built; this reflects the needs and the perspective of the content author.
- While each adaptive system has its own content requirements (based on its users), relying upon the author-imposed structure hinders these approaches from understanding the content itself and, as a result, they cannot satisfy these content requirements.
- Furthermore, for content resources that do not possess any layout structure, extracting relevant content fragments is not then feasible.
- Additionally, as these approaches can only process content resources that have a layout structure, they are limited with regard to the diversity and the volume of content resources which they can leverage and provide to adaptive systems.

Key Principle 3: The proposed service should employ novel approaches to **structure content resources without the reliance upon their original structure**. This would allow the service to incorporate a wider range of content resources regardless of its origin (closed or open) and the method used to incorporate it (user incorporation or automatic incorporation – section 2.4.4.2).

5.1.4 Content Representation

- State of the art approaches are limited in that they rely upon the traditional bag-of-words representation of content in measuring similarity between a content item (a document or a fragment) and the request (query) sent by individual users or applications (section 2.5.2).
- Even systems that use concepts in this task primarily rely upon a limited conceptual representation of content items (i.e. using one or very few concepts) (section 2.5).
- Such limited conceptual representation hinders content items from being effectively discovered and presented to the users (or applications) according to their needs and from being reused in other systems (section 2.5.2).

Key Principle 4: The service should apply **a conceptual representation of the incorporated content** to overcome the limitations of the lexical representation paradigm. Furthermore, this conceptual representation should be rich enough to cover all the possible interpretations of content.

5.1.5 Content Indexing

- Content indexing provides a formalised, simplified and machine usable representation of content contained within each resource.
- For adaptive systems that rely on closed corpus content resources, since content is structured and annotated by the content author (or a domain expert), the indexing of this content is typically manual, which is a labour-intensive task (section 2.5.2).
- For adaptive systems that rely on open corpus content resources and use document-level indexing approaches, the harvested resources are indexed in their native form as one-size-fits-all, document-level-granularity resources (section 2.5.2.1).
- This in turn limits the extent to which these resources can be modified or recomposed together.
- For fragment-level indexing approaches, the harvested resources are processed and segmented into coherent fragments based on their layout structure. The produced fragments are then indexed in a content repository (section 2.5.2.2).

- However, in these approaches, the indexing process means that the final structure of each content fragment is already built. This limits the capability of these approaches from changing such structure according to the needs of individual users or applications and makes these approaches limited in their ability to provide different levels of granularity for the indexed content.
- Additionally, both approaches (document-level and fragment-level) are limited in that they index content items (documents or fragments) based on the keywords or few concepts (usually one concept) found in each individual item.
- As mentioned in the previous section (section 5.1.4), this prevents content items from being effectively discovered and presented according to the users' needs and hinders their reusability in other systems.

Key Principle 5.a: Content needs to be **automatically indexed** at various levels of granularity regardless of the original structure that has been embedded by the author of the content resource.

Key Principle 5.b: Such indexing mechanism should **index each level of content granularity** along with a rich conceptual representation for each content item (i.e. content slices) in each level.

5.1.6 Content Discoverability

- Discovery of the indexed content involves deciding what content is most relevant to the needs or goals of current user or application. Generally, strategies for content discovery compute a measure of relevance for each content item (e.g. fragment) to the target user's model (section 2.5.3).
- In keyword-based approaches, the user model is represented as a set of *keywords* that describe the user's preferences or goals and a *keyword-based* similarity measure is computed between this user model and content items.
- However, the quality of simple keyword-level similarity techniques is not reliable as it can often retrieve content items that are not semantically related to the user model (section 2.4.4.2).
- In concept-based approaches, the user model is represented as a set of *concepts* that characterise the user's preferences or goals and a *concept-based* similarity measure is computed between this user model and the concepts covered in content items (section 2.4.4.2).

- However, as discussed in section 5.1.4, these systems mainly rely upon a limited conceptual representation of content items (i.e. using one or very few concepts). Such limited conceptual representation hinders content items from being effectively discovered and presented to the users according to their preferences or goals (section 2.5.3).

Key Principle 6.a: The process of **discovering (and providing) content items should be performed by the service using the conceptual representation** of the indexed content. The similarity between a request (sent by the adaptive system based on the user model) and a content item (a slice) should be measured based upon the conceptual representation of the request and the content item.

- Another limitation of the current state of the art approaches is that, since they rely upon the original structure of the incorporated content, they have limited control over the granularity level of content and can only provide very coarse-grained content fragments (e.g. fragmenting a document down to paragraphs, but not to sentences or phrases for example).

Key Principle 6.b: The proposed service should be able to **discover content items according to any level of granularity** specified by the adaptive system that suits its requirements.

5.1.7 Content Reusability

- Different content reusability techniques have been proposed in the literature where each technique has its own strengths and limitations (section 2.6).
- *Encapsulation* techniques showed that content encapsulation standards allow a common structure and descriptive metadata to be used across resource consumers, which supports the discovery and reuse of content resources from various origins by each individual application (section 2.6.1).
- However, with the increase in volume of content resources, the amount of manual labour required to annotate and structure these resources also increases. Furthermore, repositories produced by content encapsulation techniques can still be seen as "closed pools" of reusable content resources, since the resource publication and delivery mechanisms are specific to each repository.

- *Shared Publishing* techniques removed the need for adaptive systems to modify its content consumption mechanism to each individual repository and improved the accessibility of content resources from the different repositories as if they are from a single unique repository (section 2.6.2).
- However, both encapsulation and shared publishing reuse mechanisms do not involve the modification of the original resources themselves. Although reusing a content resource without modifying it can potentially include reusing it for purposes not originally planned by its author, the inability to modify a content resource limits the range of purposes for which it could be reused.
- *Content Modification* techniques started to apply modification to the original resources to increase the reuse capabilities of a content resource (section 2.6.3).
- However, relying on the original structure of the content resource (HTML structure or paragraphs) to modify it, limits the range of purposes for which it could be reused.
- Additionally, since these approaches modify content resources and produce content fragments before a request is done, the produced fragments are considered static content items which restricts the potential scenarios in which such content items can be reused. This, in turn, makes these approaches limited in responding to the different potential forms of requests.

Key Principle 7.a: The proposed service should **automatically modify** the incorporated content resources to produce content slices at all possible levels of granularity of each individual content resource. This would allow the creation of a slice on-the-fly according to the characteristics of an arbitrary request by individual adaptive systems.

- A common characteristic of all the aforementioned approaches is that the reusability of a content item within different adaptive systems depends primarily upon the descriptive metadata they attach to content items which represents the content publication and delivery mechanism of each approach. This descriptive metadata is used to select appropriate content where there may be many candidate content items available to fulfil a user or system requirements.
- However, as mentioned in section 5.1.4, these techniques provide limited capabilities to capture the conceptualisations associated with content resources and hence they offer limited metadata information about content items. This, in turn, limits the scenarios in which each individual content item can be reused.

Key Principle 7.b: To increase the potential scenarios in which content items can be **reused**, the proposed service should **provide a generic content publication and delivery mechanism** to allow adaptive systems to easily acquire content items without the need to adjust their content acquisition mechanism. This means that a rich descriptive metadata information should be attached to each content item. This metadata information should have all possible conceptual representations of the content item.

5.1.8 Summary

This section discussed the influences derived from the analysis conducted in the state of the art review (Chapter 2). The analysis revealed the limitations with respect to the underlying content production approaches currently used to supply content to adaptive systems. The analysis also revealed the potential enhancement that can be applied to the content production (and supply) process to enhance the discoverability of the appropriate content items as well as the reusability of content for different adaptive systems. This section summarised these influences and key principles made with respect to the prototype service to be developed for this research. The next section provides a description of design requirements, derived from the key principles enunciated in this section, of a service which enables the automated harvesting, structuring, customisation and reuse of open and closed corpus content resources for consumption within adaptive systems based on content's conceptual representation.

5.2 CROCC Architecture

The main purpose of the CROCC service is to provide adaptive systems with textual content slices that are tailored to the needs and requirements of each individual system. The CROCC service is designed using a flexible architecture that allows for plugging-in, removing, enabling, or disabling alternative components or algorithms at runtime as well as design time of the service (*Key Principle 1.b*). The CROCC service is offered as an intelligent content-supply framework, which consists of the following modules:

1- Content Harvester
2- Content Pruner
3- Structure Builder
4- Slice Indexer
5- Content Repository
6- Slice Selector

Figure 5.1 depicts the architecture of the CROCC service. This architecture is designed as a pipeline where the components can be run successively, in the numerical order specified in the figure.

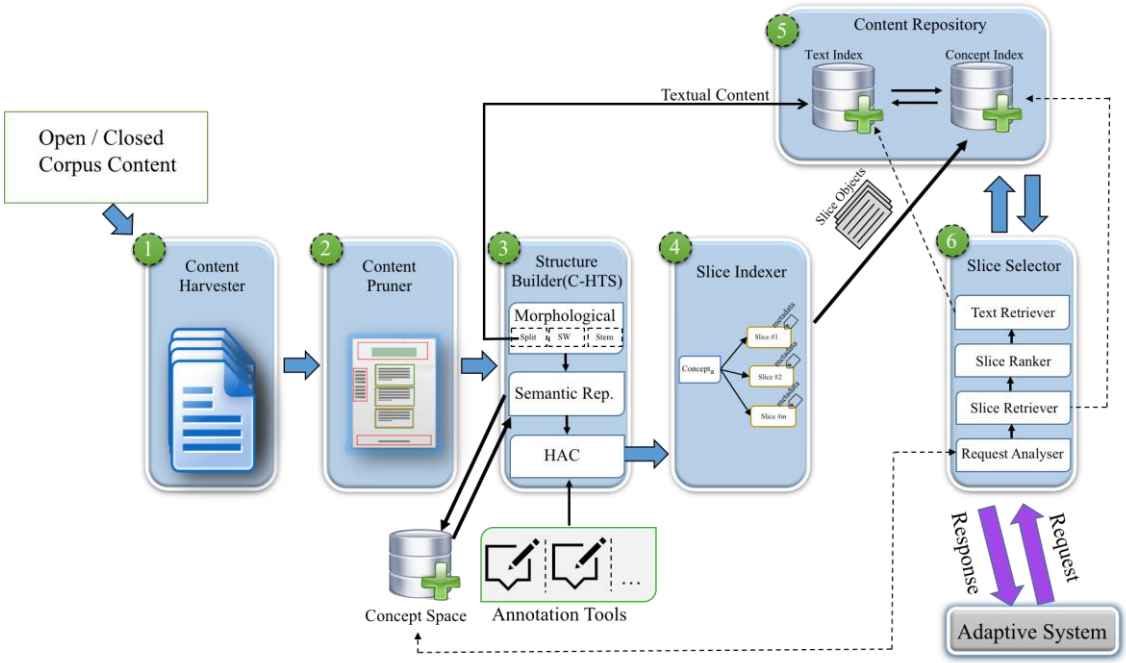


Figure 5.1 The CROCC service architecture

5.2.1 Content Harvester

The first module in the CROCC service is the content harvester. The purpose of this module is to automatically acquire content resources based on the content requirements of the adaptive system (*Key Principle 1.a*). These requirements include: domain area (e.g. health, computer science, finance, etc.), a set of URLs the adaptive system needs content to be harvested from and/or set of closed corpus content resources (e.g. a book in an educational adaptive system). Since CROCC can operate on both closed and open corpus content resources, the harvester mainly provides two different content harvesting modes: *closed corpus harvesting* and *open corpus harvesting*. **For closed corpus harvesting**, and as discussed in section 2.5.1, since content resources are known to the adaptive system (e.g. specified by a domain expert or incorporated by its users), the Content Harvester provides an interface for the adaptive systems to allow them to transfer (upload) these content resources to the CROCC service to be processed and indexed as content slices. **As for open corpus harvesting mode**, since content resources are not known to the adaptive system, they need to be discovered and retrieved to be processed by the CROCC service; hence, the main purpose of the open corpus harvesting mode is to *automatically* identify and incorporate relevant resources from the web which relate to a topic from the

specified domain area. This technique mainly focuses on identifying web resources relevant to a particular topic and harvesting them in their native form (HTML pages). Since the challenge of identifying web resources relevant to a particular topic is a very well documented and established field of research of its own (see section 2.5.1), the process of correctly identifying relevant web pages is considered out-of-scope of this thesis. Thus, existing third-party IR systems or focused crawling techniques can be utilised instead as a callable service in this module. Examples of such systems are: Bing Custom Search[4], Openwebspider[5], Heritrix[6], 80Legs[7] and WebSphinx[8]. These systems enable the community to perform a variety of crawling tasks (from classic crawling to focused crawling).

5.2.2 Content Pruner

As discussed in section 5.1.3, the CROCC service should employ novel approaches to build a structure out of textual content resources regardless of the original structure of these resources (*Key Principle 3*). However, content resources usually come in a range of different formats, where each format type has its own structure. For example, open corpus resources available on the web are usually formatted as HTML pages with deep nested structures that contain auxiliary content fragments, such as: headers with navigation menus, footers with contact and corporate information, or sidebars with advertisements (Viveros-Jiménez et al., 2018; Vogels et al., 2018). In order to build a structure out of a content resource, such content fragments are unwanted or not useful (Levacher et al., 2009). Hence, these content fragments must commonly be discarded before building the structure for the content resource.

Content Pruner hence refers to the ability to identify such unnecessary fragments within a content resource to remove them and convert the content resource into a plain text file. Figure 5.2 depicts the process of pruning content resources and removing the unwanted content fragments.

In contrast to state of the art content-supply approaches, the CROCC service does not rely upon the original structure of the content resource, rather, it prunes the resource and converts it into a plain text in order to build a structure out of it based on the conceptual coherency between its constituents.

[4] https://azure.microsoft.com/en-gb/services/cognitive-services/bing-custom-search/[Accessed May 2, 2018]
[5] http://www.openwebspider.org [Accessed May 2, 2018]
[6] https://webarchive.jira.com/wiki/display/Heritrix/Heritrix [Accessed May 2, 2018]
[7] http://www.80legs.com/ [Accessed May 2, 2018]
[8] https://webarchive.jira.com/wiki/spaces/Heritrix [Accessed May 2, 2018]

The task of pruning a web page is called content fragmentation[9]. This task is a very well documented and established field of its own where different approaches have attempted to identify structurally coherent fragments of a web page based on its HTML layout structure (Fang et al., 2018; Zeleny et al., 2017). Content fragmentation techniques are increasingly used as preliminary processing step for different tasks such as web mining (Alassi and Alhajj, 2013), web search (Moura et al., 2010) and web page classification (Bing et al., 2014).

As CROCC can operate on closed and open corpus content resources (*Key Principle 3*), designing it using a flexible pipeline architecture enables the content pruner (and other modules) to be easily replaced based on the format of the desired content resources.

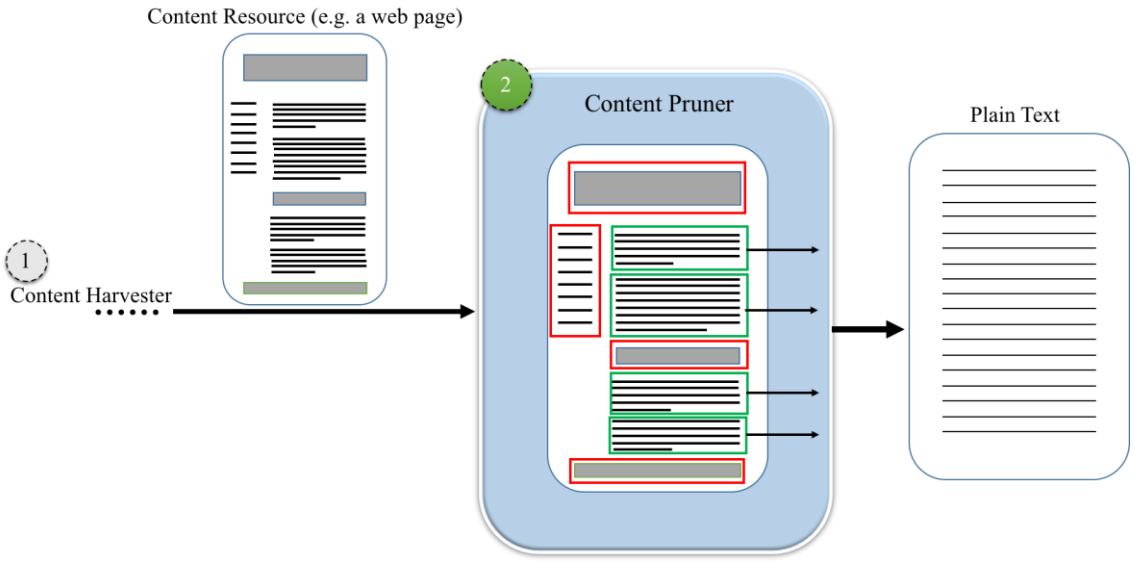


Figure 5.2 Removing the unnecessary content fragments by the Content Pruner

5.2.3 Structure Builder

After pruning the harvested content resources and converting them into plain text documents, a hierarchical structure is built for each resource (*Key Principle 2 & Key Principle 3*) based on its conceptual representation (*Key Principle 4*). To achieve this task, the Structure Builder module utilises the C-HTS algorithm proposed in Chapter 4 of this thesis. In this module, and for each resource, text is processed to be split into sentences and to remove stopwords as they are generally assumed to be of less, or no, informational value. The remaining terms are then stemmed and converted to their linguistic root form (section 4.4.1). A Hierarchical Agglomerative Clustering (HAC) approach is then applied

[9] In the literature, content fragmentation is also referred to as structural fragmentation, region extractors, and page segmentation.

on text blocks to successively agglomerate blocks that are semantically coherent and build a text structure (section 4.4.3). As discussed in section 4.6.1, the size of the elementary units for C-HTS is one sentence.

In each iteration of C-HTS, for each text block (one or more sentences), and for each term in that text block, the term is mapped to a vector of concepts from the underlying concept space (section 4.3). The semantic relatedness between two (adjacent) blocks is calculated as the cosine similarity between the centroid of the vectors representing the individual terms in each block using Equation 4.2 (section 4.3.1). The algorithm successively agglomerates text blocks that are deemed to be semantically related to each other, thus forming a text structure. C-HTS uses HAC because it is a bottom-up clustering approach. The idea behind using a bottom-up approach in building the structure of content is that it starts from the smallest clusters (sentences), that are considered the lowest granularity level of content, and then builds the content structure by successively merging the semantically coherent clusters. This way of building the content structure allows the production of fine grained content *slices* that are useful to support a variety of content discoverability and reusability methods as it provides different levels of granularity for the underlying content (*Key Principle 2*).

Figure 5.3 shows a sample of the output of one iteration (first iteration) of the Structure Builder. In this iteraton, the Structure Builder generates *slice objects* of the target content resource. Each slice object contains a set of concepts (the weighted centroid vector of concepts, section 4.3.1) along with the relevancy score of each concept with this slice. This score quantifies the strength of association between the concept and the slice. Additionally, a descriptive metadata information is attached to each slice object. This metadata describes what textual content (i.e. sentences) in the target document is covered by this slice, the document id and the size of the slice.

The metadata layer attached to each slice can include a variety of metadata information that gives more description to each slice. This is due to the flexibility of building the structure of a content resource using C-HTS. With each iteration in the algorithm and in each level, additional annotations can be easily attached to each generated slice using the approperiate annotation tool. For example, for each generated slice (one or more sentences), a *reading difficulty* score can be measured and attached to the slice. This score identifies the difficulty level associated with comprehending a piece of content by analysing its text. This would help, for example, in response to a request sent by an

educational adaptive system when it requires a slice of text with a specific reading dificulty according to its user's level.

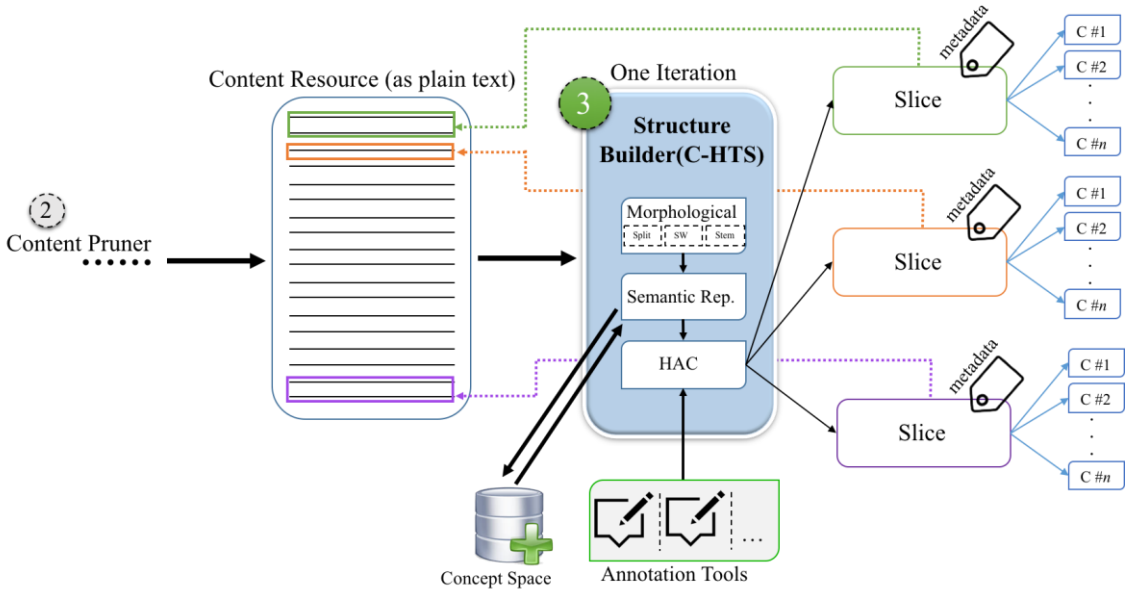


Figure 5.3 A sample of the output of one iteration of the C-HTS algorithm in the Structure Builder module

Another metadata element that could be added to the metadata layer of the slice is the pedagogical annotations of an eLearning course to provide pedagogically meaningful learning experiences. For each slice produced by the Structure Builder, automatic pedagogical annotation algorithms could be applied to categorise slices into introduction, description, quiz, explanation, example and other pedagogically meaningful concepts (Labutov et al., 2017; Sathiyamurthy and Geetha, 2011; Wang, 2008). Such metadata information could be used to identify and retrieve the content slices available from the Content Repository (see section 5.2.5) that match as closely as possible the content requirements requested by an adaptive system.

5.2.4 Slice Indexer

The main task of the Slice Indexer is to index content slices produced by the Structure Builder. These slices represent content items produced at all levels of granularity of a content resource (*Key Principle 5.a*). This process includes the indexing of the slice objects produced at each level of the hierarchical tree by the Structure Builder (*Key Principle 5.b*).

In the Structure Builder module and for each iteration of C-HTS, the Slice Indexer recieves the produced slice objects in each level. The Slice Indexer starts to index each

slice object in an inverted index in the Content Repository (see next section). In this indexing process, each concept in each slice object, is mapped to a list of other slice objects in which this concept appears, along with the descriptive metadata information attached to the slice. Thus, each concept appearing in documents of the harvested corpus can be seen as triggering each of the slice objects it points to in the inverted index. Figure 5.5 shows how the Slice Indexer maps each concept to slice objects associated with it.

It is worth mentioning that the Slice Indexer does not index the textual data of each produced slice. Rather, it indexes the slice object that describes the content of the slices. Thus, a slice object is not, in fact a slice, but is just a metadata description of a slice. Indexing the textual data of each harvested resource is described in section 5.2.5.2.

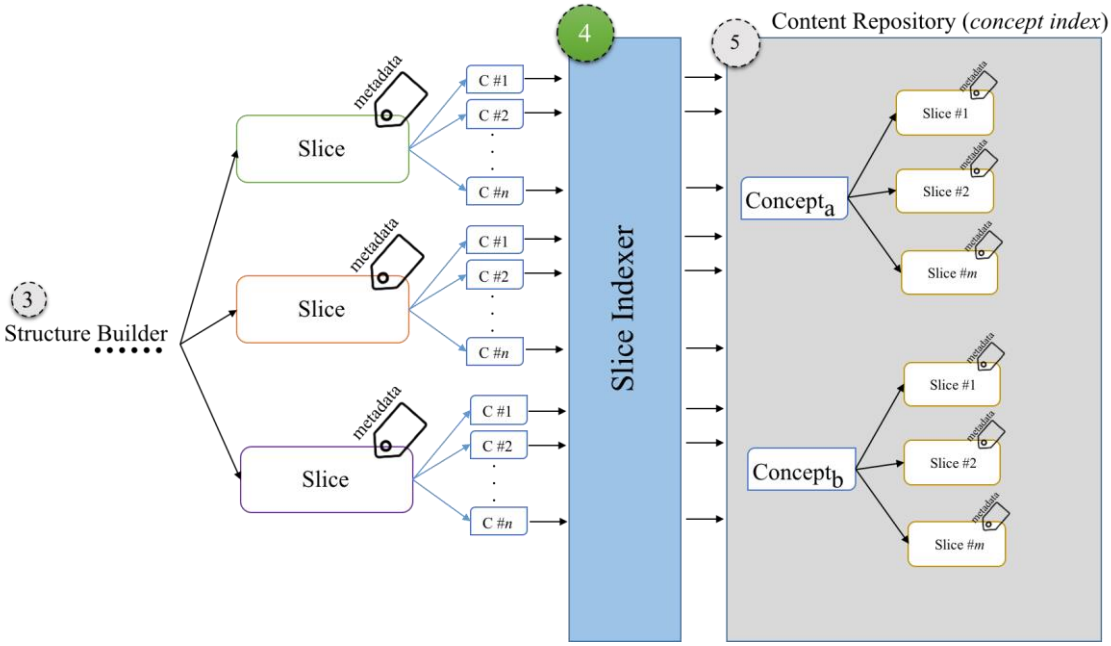


Figure 5.4 Illustration of how the Slice Indexer maps a concept to slice objects associated with it

5.2.5 Content Repository

As discussed in the previous section, for a given document, in each iteration of C-HTS in the Structure Builder, the Slice Indexer maps each concept in each slice object to the set of slice objects associated with this concept. In order to facilitate the discoverability of content slices, the Slice Indexer sends these slices to the Content Repository to be indexed. The Content Repository consists of two indices: Concept Index and Text Index. Both are standard IR inverted indexes that store data in a key-value indexing fashion.

5.2.5.1 Concept Index

The main task of this index is to store concepts and their relevant slice objects produced by the Slice Indexer. It is a standard IR inverted index that stores data in a key-value indexing fashion where the *key* is the id of the concept as it appears in the underlying concept space and the *value* is the set of slice objects associated with that concept. Each slice object contains metadata that describes the slice. Figure 5.5 shows an example of a concept index in the Content Repository. In this example, a concept with id 50549 is mapped to a set of slice objects from the harvested corpus that this concept is associated with. Each slice object contains the following metadata information:

- *Document ID*: the id of the document (in the harvested corpus) where this slice comes from.
- *Slice Range*: the range of sentences (textual content) that this slice covers in the document. The slice range indicates the start index and the end index of the sentences that it covers in the document. For example, in Figure 5.5, slice with *Slice Range* = 1 means that this slice consists of one sentence with index = 1 (first sentence in the document). Also, the slice with *Slice Range* = 5-8 means that this slice consists of 4 sentences starting from sentence 5 and ending at sentence 8 in the document. This content identification mechanism saves space and allows the Slice Selector (see section 5.2.6) to easily discover and retrieve relevant slices according to the requirements of different individual adaptive systems. What this means is that CROCC is space-efficient in that it does not store the extracted fragments as a chunk of text, but rather stores fragment metadata that acts as pointers to various parts of the text.
- *Score*: the relevance score of the concept for this slice. This score identifies the order of each slice in the list of slice objects associated with the concept. In other words, slice objects in this list are ranked in *descending* order based on their relevance for this concept (this score is calculated as described in section 4.3.1).
- *Size*: the number of sentences in the slice. Although the number of sentences can be inferred from the *Slice Range*, this number is stored in the index in order to speed up the search process when we want to retrieve slices of specific size (e.g. we only want slices that contain five sentences). This size can be mapped to the adaptation requirements of an adaptive system. For example, it can be mapped to the *reading time* of the user in the request (Lawless et al., 2015).

As discussed in section 4.4.1, C-HTS splits text into sentences. Consequently, the slice size is represented as the number of sentences in that slice. However, since the morphological analysis process in C-HTS walks through each term in each sentence (removing stopwords and stemming), a slice size can also be represented in terms of the number of tokens that the slice covers. This in turn gives more control over content granularity and gives the CROCC service the ability to produce different compositions of slices to meet the adaptation requirements of individual adaptive systems.

- *Other metadata annotations*: as discussed in section 5.2.3, the flexibility of the C-HTS algorithm while building the document structure allows the integration of different annotation tools that can annotate each slice with different information. Such metadata annotations can be easily added to the metadata information of the slice. Examples of such metadata annotations are reading difficulty and pedagogical concepts.

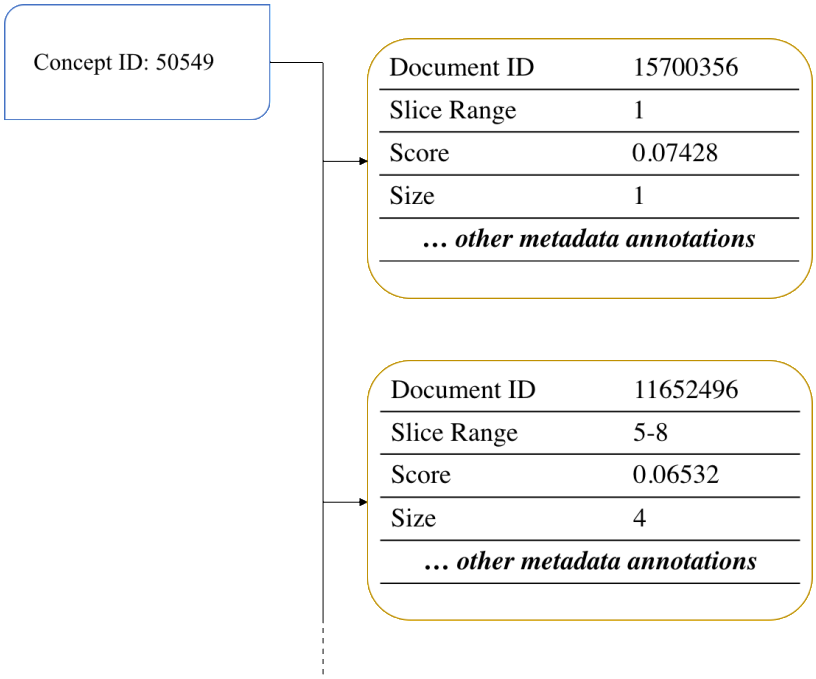


Figure 5.5 An example of the concept index in the Content Repository

This index, hence, provides a concept-based content publication and delivery mechanism that allows adaptive systems to easily acquire content slices. All it needs is that the adaptive system employs the proper technology to call the CROCC service or make some minimal adjustments to their content acquisition mechanism. This in turn increases the potential scenarios for which the incorporated content resource could be reused (*Key Principle 7.b*).

5.2.5.2 Text Index

The second index in the Content Repository is the Text Index. The main task of this index is to store the textual content of the harvested resources. When a content resource (from closed or open corpus) is harvested by the CROCC service using the Content Harvester, the Content Pruner removes the unnecessary content fragments and converts it into plain text. This plain text is then processed by the morphological analysis phase of the C-HTS algorithm in the Structure Builder module. The first step in this phase is to split text into sentences. Since the final output of the CROCC service is a slice of textual content, the produced sentences from the morphological analysis phase are sent to the Text Index to be indexed as a list of sentences. This list is used later to generate the textual content of the selected slice. Note that, stopwords removal and stemming are not performed on the text in this index as it will be used to reassemble the sentences in the slice (see next section). Figure 5.6 illustrates how the document sentences are stored in the Text Index after the morphological analysis phase in C-HTS.

Similar to the Concept Index, the Text Index is a traditional IR inverted index that stores data in the key-value fashion where the *key* is the document id and the *value* is a list of sentences that this document contains. Figure 5.7 depicts a sample of a document indexed in the Text Index.

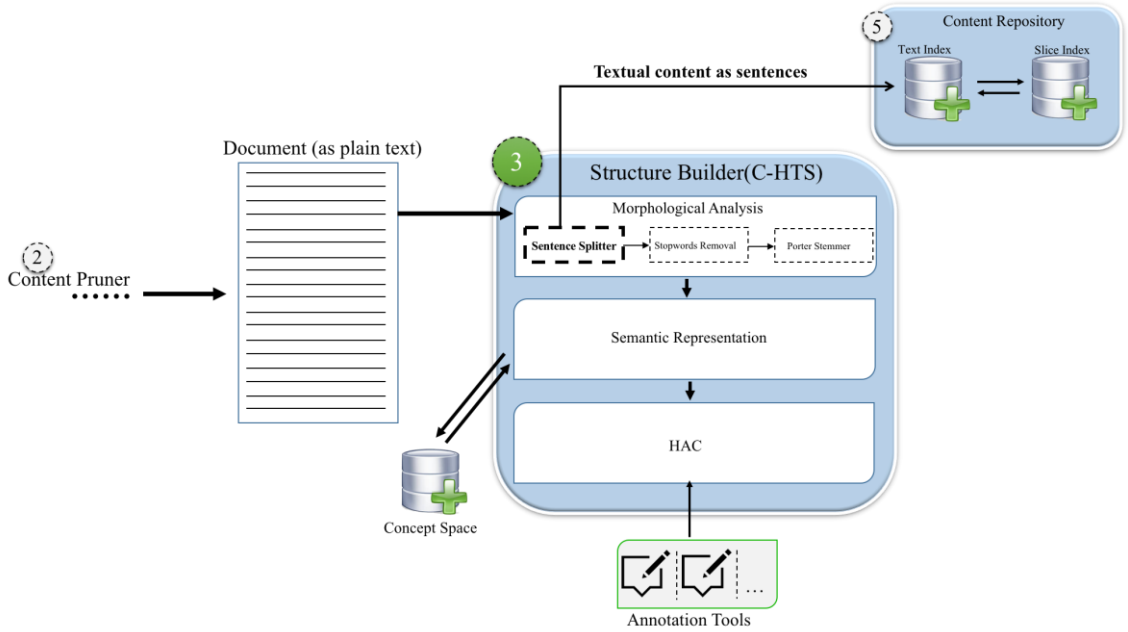


Figure 5.6 Document sentences stored in the Text Index after the morphological analysis phase in C-HTS

Document ID: 11652496

#	Text
1	We now develop a scheme in which, given a query, every web page is assigned two scores.
2	One is called its hub score and the other its authority score.
3	For any query, we compute two ranked lists of results rather than one.
4	The ranking of one list is induced by the hub scores and that of the other by the authority scores.
5	This approach stems from a particular insight into the creation of web pages, that there are two primary ……
6	By a broad topic search we mean an informational query such as "I wish to learn about leukemia.
7	There are authoritative sources of information on the topic; in this case, the National Cancer Institute's ……
8	We will call such pages authorities; in the computation we are about to describe, they are the pages that …….
…	…….

Figure 5.7 A sample of a document indexed in the Text Index

5.2.6 Slice Selector

Once slice objects and textual content have been indexed, the CROCC service becomes ready to provide content slices in response to requests sent by adaptive systems. The Slice Selector is considered the interface of the CROCC service with the adaptive systems. Its main task is to receive the request, process it, and retrieve the slice that best matches the request. Figure 5.8 depicts how the Slice Selector module works.

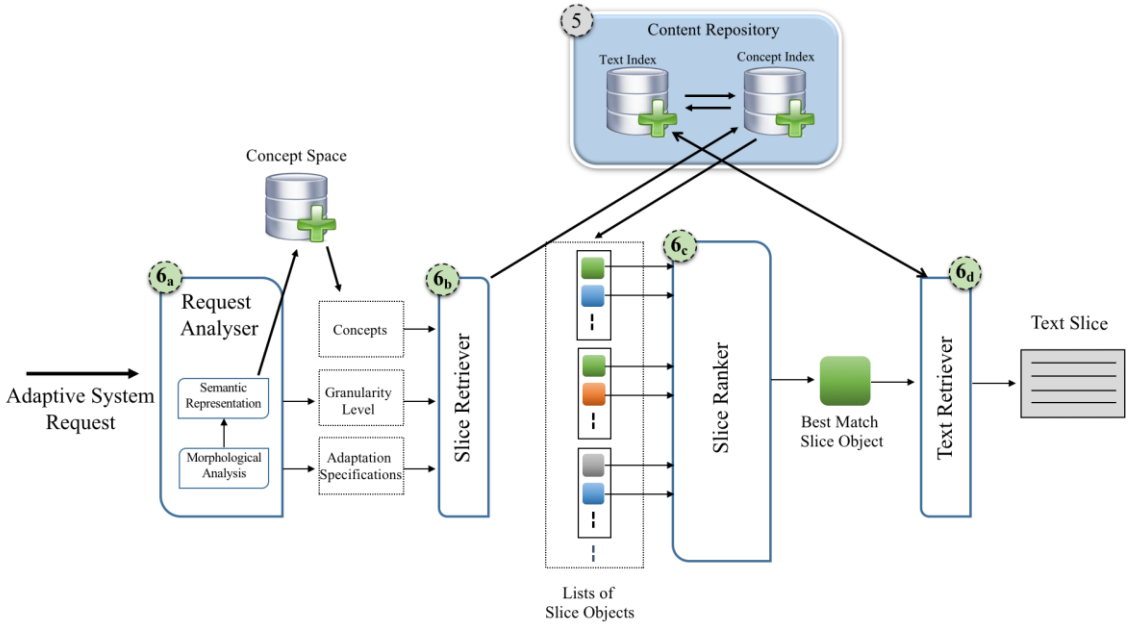


Figure 5.8 Illustration of how the Slice Selector module works

The Slice Selector consists of four main sub-modules:

a) Request Analyser: upon receiving a request, the request analyser starts to convert the query (the textual content) in the request into an ESA concept vector (section 4.3.1). The representation method of the query is identical to the one by which slices are represented at index time. First, it starts to apply the morphological analysis to the query to remove stopwords and then stem the remaining terms (section 4.4.1). Each term is then mapped to a vector of concepts from the underlying concept space used in the indexing process. The query is then represented as the centroid vector of all concepts associated with the query terms. Concepts in this centroid vector are ranked by their relevance to the query (using Equation 4.1). Figure 5.9 depicts a sample of the centroid vector of three concepts with their relevance scores to the query.

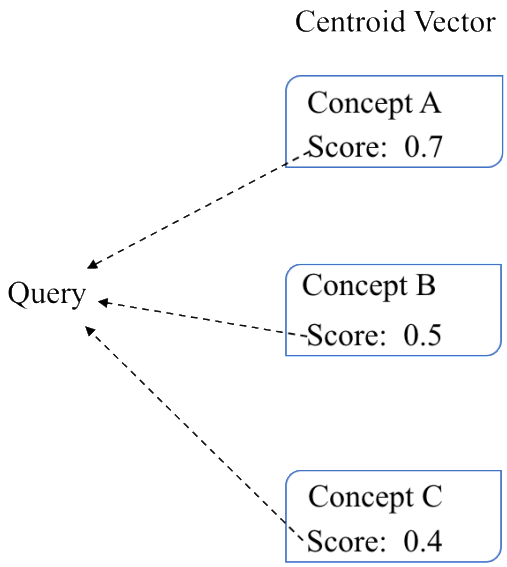


Figure 5.9 A sample of the centroid vector of three concepts with their relevance scores to the query

As discussed earlier in section 5.1.6, a common limitation of the state of the art approaches is that they have limited control over content granularity and they can only provide very coarse-grained content fragments. This is due to their reliance upon the original structure of the content resource. Since the Structure Builder module structures content in different levels of granularity, the CROCC service allows the adaptive system to specify its desired level of granularity for the requested slice; this is of course based on the user's needs (*Key Principle 6.b*). The level of granularity can be specified in terms of number of sentences or number of tokens. This in turn gives the adaptive system more control over content granularity and hence overcomes the limitations of the state of the art approaches.

In addition to the level of granularity of the requested slice, the CROCC service allows the adaptive system to specify different adaptation specifications that suit its needs (section 2.4.4). For example, a *reading difficulty* specification can be attached to the request to specify the difficulty level associated with comprehending the textual content of the slice. Additionally, a specification of *pedagogical concept* can be attached to the request to specify the category to which the requested slice should belong (e.g. Introduction, Explanation, etc.).

b) Slice Retriever: after building the centroid vector of concepts from the query and identifying the level of granularity, along with the adaptation specifications attached to the request, the slice retriever starts to query the Concept Index using this information. In this process, for each concept in the query centroid vector, a list of slice objects associated with that concept is retrieved such that: (1) they adhere to the specified level of granularity; and (2) they match the adaptation specifications of the request. Figure 5.10 shows an example of the returned lists of slice objects[10] associated with the three concepts depicted in Figure 5.9. Note that, as mentioned in section 5.2.5.1, in each list, the slice objects are sorted in descending order of their relevance score with the concept.

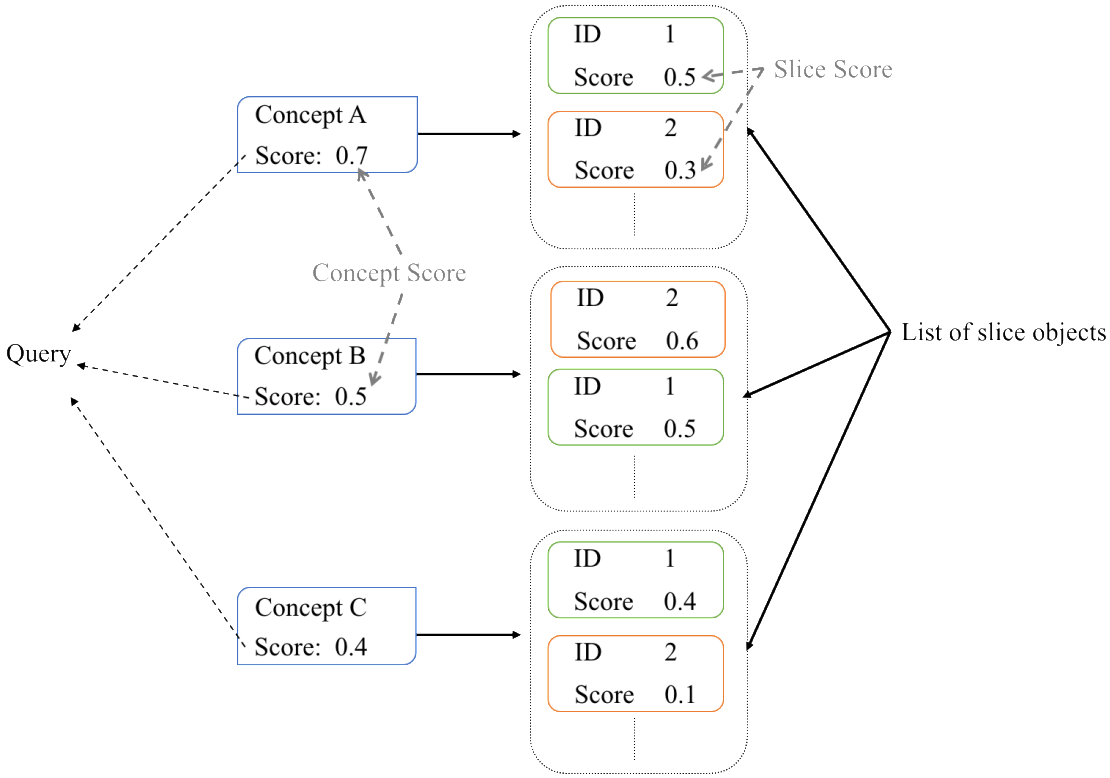


Figure 5.10 Example of the returned lists of slices associated with the three concepts in Figure 5.9

[10] Other metadata information (e.g. size, document id, etc.) were omitted from this figure for simplicity.

c) Slice Ranker: the ranker takes input in the form of multiple lists of slice objects (as produced by the previous step). In order to select the slice that best matches the request, the slice ranker starts to assign a weight to each slice based on two main scores (Figure 5.10):

1- Slice Score: the relevance score between the slice object and the concept.
2- Concept Score: the relevance score between the concept and the query.

Using both scores, the final weight of a slice object is calculated as follows::

$$S^i_{weight} = \sum_{j \in \vec{m}} S^i_{cj} * C^j_q \qquad 5.1$$

where S^i_{cj} is the *slice score* of slice object i with concept cj, C^j_q is the *concept score* of concept j with query q, and $\vec{m}$ is the centroid vector of all concepts associated with the query terms.

This score is the accumulation of the concept weight in the centroid vector and the weight of the slice object in the list associated with that concept. Slice objects are then ranked by their final score and the top ranked slice object is selected.

d) Text Retriever: since the final output of the CROCC service is a slice of textual content, the metadata information associated with the top ranked slice object is used to select the slice's text from the Text Index. Using the "*Document ID*" and the "*Slice Range*" parameters of the metadata information attached to the slice, the text retriever queries the Text Index to retrieve the textual content of the slice. This textual content is then returned as the response to the request sent by the adaptive system.

This process of automatically identifying and generating a content slice demonstrates how the CROCC service is capable of producing a slice on-the-fly according to the characteristics of a request from an adaptive system (*Key Principle 7.a*). This in turn enhances content reusability by increasing the potential scenarios in which the incorporated content resource could be reused (*Key Principle 7.b*).

5.3 Adhering to the Key Principles

The previous section (section 5.2) described the design aspects of the CROCC service. This section discusses how the proposed design of the CROCC service adheres to the key

principles derived from the state of the art influences. With respect to these key principles, the way that CROCC is designed has demonstrated the following:

1- CROCC is provided as a service that:

   a. Allows the *automatic* harvesting and incorporation of content resources using the Content Harvester (*Key Principle 1.a*)

   b. Has a usefully flexible architecture (*Key Principle 1.b*)

2- Using the Structure Builder, the service structures the harvested content resources and converts them into fine grained content packages (slices) (*Key Principle 2*)

3- The service prunes the incorporated resources using the Content Pruner and structures each individual content resource without relying on the original structure. Thus, the service allows the incorporation of any relevant content resource regardless of its source (closed or open) and regardless of the method used to incorporate that resource (user incorporation or automatic incorporation) (*Key Principle 3*).

4- The service builds the structure of a resource based on the conceptual representation of its content using a concept space built from a massive knowledge base. Such a knowledge base is based on human cognition and has no limits on domain coverage and conceptual granularity. This in turn allows the service to represent the textual content of resources with a rich conceptual representation that covers all possible interpretations of the content (*Key Principle 4*).

5- Using this rich conceptual representation of content resources:

   a. The service automatically indexes slices in all levels of granularity of a content resource using the Slice Indexer (*Key Principle 5.a*)

   b. Each level of content granularity is indexed in the Concept Index along with all possible concepts associated with each individual content slice in that level (*Key Principle 5.b*).

6- The Slice Selector module of the service processes the adaptive system request to:

   a. Build a conceptual representation for the query of the request in order to discover content slices based on that representation (*Key Principle 6.a*)

b. Identify the required level of granularity and other adaptation specifications required by the adaptive system (*Key Principle 6.b*)

7- All the aforementioned characteristics of the CROCC service demonstrate that:

a. As the service automatically processes content resources in a manner that allows the production of content slices at any level of granularity, the produced slices are considered dynamic content items. This, therefore, allows the production of content slices on-the-fly based on the adaptive system requirements (*Key Principle 7.a*).

b. As the service enriches each content slice with descriptive metadata information that has all the conceptual representations of the slice, it provides a generic content publication and delivery mechanism that allows adaptive systems to easily acquire content items without the need to adjust their core implementation (e.g. their adaptation mechanism). This in turn increases the potential scenarios where the incorporated content resource could be reused (*Key Principle 7.b*).

5.4 CROCC Implementation

This section describes a prototype implementation of the CROCC service proposed in this chapter. This implementation provides a system for evaluating the collective effectiveness of the various components that make up the CROCC service. The implementation served as the basis for the evaluation reported in Chapter 6.

The CROCC prototype is fully implemented in the Java Programming language. As discussed in section 5.2, CROCC is designed as a flexible architecture in order to accommodate future improvements in the state of the art of relevant dependent fields of research. Hence, the implemented prototype allows for plugging-in, removing, enabling, or disabling alternative components or algorithms at runtime as well as design time. The following subsections describe the technical implementation of each component of the prototype.

5.4.1 RESTful Web Service

In order to make CROCC accessible by any adaptive system, the CROCC prototype was incorporated as part of a RESTful web service using the Java Spring framework[11]

[11] https://spring.io/

(Nguyen, 2018). The prototype was incorporated as a service, so it can accept HTTP GET or POST requests from any adaptive system and respond with a JSON[12] representation of a content slice; thus, confirming to standard and common RESTful web services found on the web.

As CROCC is a RESTful web service it is easy to access and does not require extra development effort in order to be integrated with. All it needs is that the adaptive system employs the proper technology to call the service or make some minimal adjustments to their content acquisition mechanism. This, therefore, allows the adaptive systems to easily acquire content slices without the need to change their core implementation (e.g. their adaptation mechanism).

5.4.2 Content Harvester

As discussed in section 5.2.1, the Content Harvester provides two different harvesting techniques: *closed corpus harvesting* and *open corpus harvesting*. For closed corpus harvesting, since the CROCC prototype is implemented as a RESTful webservice, the service allows the adaptive system to send its content resources (e.g. PDF documents, Word files, etc.) using an HTTP POST requests through the Request Coordinator component (see section 5.4.8).

For the open corpus harvesting technique, since the process of correctly identifying relevant resources is not within the scope of this thesis, a third-party IR system is employed to carry out this task. Among the various available systems, Microsoft's Bing Custom Search[13] was chosen for this implementation due to its availability, scalability, ease of configuration and good performance[14]. Bing Custom Search is a search solution that leverages the powerful capabilities of the Bing search engine[15,16] while allowing users to customise their search experience through an easy-to-use API. The API allows users to specify which domains, subsites, or webpages to surface results from. This enables users to build a tailored search experience for different topics.

Therefore, the open corpus harvesting technique simply consists of a Java wrapper component around the Bing Custom Search API. This wrapper calls the API and specifies:

[12] JavaScript Object Notation
[13] https://azure.microsoft.com/en-gb/services/cognitive-services/bing-custom-search/
[14] We have tried the Bing Custom Search previously in (Lawless et al., 2015) and its performance was very reliable.
[15] https://www.bing.com/
[16] Formerly known as MSN Search until September 2006, and as (Microsoft) Live Search until May 28, 2009

(1) the search topic; (2) the domain to focus on (e.g. health, finance, computer science, etc.); and (3) if required by the adaptive system, the desired subset of target webpages (e.g. Wikipedia pages only). The response from the API is a JSON object that contains URLs of pages that are retrieved from the search process. These pages are then downloaded in their native form as HTML documents and sent to the Content Pruner module of the service.

5.4.3 Content Pruner

As discussed in section 5.2.2, the main purpose the Content Pruner is to remove the unnecessary content fragments from the incorporated content resources and convert them into plain text files. Developing a new approach to content pruning is not within the scope of this thesis, therefore, a third-party system is employed to carry out this task.

Since the CROCC service can operate on closed and open corpus content resources, two different content pruners were employed: *HTML pruner* and *Non-HTML pruner*. The HTML pruner is mainly used for the harvested HTML web pages from the web. Its main task is to remove the unnecessary fragments of a web page (e.g. headers with navigation menus, footers, etc.) using its layout structure. Among the wide variety of systems that can be used to do this task, and for the purpose of the experiment conducted to evaluate the performance of the developed prototype (Chapter 6), the Java Wikipedia Library (JWPL)[17] (Ferschke et al., 2011) was used. The library contains a mark-up parser that can be used to analyse the contents of a Wikipedia page, identify the deferent regions of the page (e.g. sections, infobox, references, etc.) and extract textual content from each page.

The Non-HTML pruner, on the other hand, is used to prune content resources that are not in HTML format. These content resources can be from closed or open corpus content and can be in different formats, e.g. PDF, Text, Word, etc. Since the CROCC service is designed using a flexible pipeline architecture, this flexibility enables different implementations of the Non-HTML pruner to be switched in and out depending on the format of the desired content. In this research, and for the purpose of the experiment conducted to evaluate the performance of the developed prototype (Chapter 6), the PDFX[18] (Constantin et al., 2013) was employed for this task. PDFX is a role-based system that extracts the logical structure of documents in PDF form.

[17] https://dkpro.github.io/dkpro-jwpl/ [Accessed May, 2018]
[18] http://pdfx.cs.man.ac.uk/

5.4.4 Structure Builder

This module utilises the C-HTS algorithm proposed in Chapter 4 of this thesis. In order to comply with the CROCC service prototype implementation, the C-HTS algorithm was entirely implemented in the Java programming language[19]. C-HTS uses a concept space to semantically represent the pruned content resource. This concept space was built from Wikipedia as discussed in section 4.3.1. In this thesis, a Wikipedia snapshot from April 2017 was processed in order to build the concept space. To process the snapshot, we used the EasyESA tool (Carvalho et al., 2014). EasyESA is a Java open source tool that provides an Explicit Semantic Analysis (ESA) infrastructure. The tool processes each article in the Wikipedia dump and indexes terms and their associated concepts in a MongoDB[20] database (see section 4.3.1 for more details on how this process is done). This database is then used as the concept space for C-HTS.

In the morphological analysis phase of C-HTS (section 4.4.1), the Apache OpenNLP library[21] was adopted to implement the three components: sentence splitter, stopwords removal and Porter stemmer. This library is used to do the same tasks in the EasyESA tool while building the concept space from Wikipedia. For the semantic representation and relatedness phase in C-HTS (section 4.4.2), we used the EasyESA tool. The tool provides an interface that calculates the semantic relatedness measure between two terms (or two sentences) using the underlying concept space. For the HAC algorithm (section 4.4.3), a java implementation of the algorithm was carried (within C-HTS implementation) for the purpose of this research.

As discussed in section 5.2.5.1, different annotation tools can be integrated with the Structure Builder in order to add extra metadata information to the produced slices. The type of information required depends primarily upon the adaptive system making the request. In other words, the annotation tool used and metadata generated is adaptive system dependant. Since designing and building an adaptive system is out-of-scope of this thesis (section 1.1), incorporating such metadata information and evaluating their influences on the CROCC service is reserved for future work (section 7.3.3).

[19] https://github.com/bayomim/C-HTS
[20] https://www.mongodb.com/
[21] https://opennlp.apache.org/

5.4.5 Slice Indexer

The main task of this module is to index the produced slice objects obtained in each level of the structure produced by C-HTS and index them in the Content Repository. Since this module is specifically designed to work with C-HTS, a Java implementation of this module was carried out for the purpose of this research.

5.4.6 Content Repository

As discussed in section 5.2.5, the Content Repository consists of two standard IR inverted indices (concept index and text index) that store data in a key-value indexing fashion. Since the concept space built for this thesis was indexed in a MongoDB database, and for system compatibility and performance, both indices were also built as standard collections[22] in a MongoDB database.

5.4.7 Slice Selector

This module comprises four sub-modules, namely: Request Analyser, Slice Retriever, Slice Ranker and Text Retriever. In the Request Analyser and as mentioned in section 5.2.6, the representation method of the query in the request is identical to the one by which slices are represented at index time. Since the indexed slices are produced by the Structure Builder, the morphological analysis and the semantic representation phases of the Request Analyser are the same as in the Structure Builder. Regarding the other three sub-modules, a Java implementation was carried out for the purpose of this research according to the design specifications of each module enunciated in section 5.2.6.

5.4.8 Request Coordinator

As mentioned in section 5.2.1, the Content Harvester allows adaptive systems to specify the domain that they want content in and/or a of URLs they want content to be harvested from. Additionally, the harvester provides the capability of uploading content resources provided by the adaptive system. This means that there are two different types of requests that the CROCC service can accept from the adaptive system: *content preparation request* and *slice retrieval request*. The content preparation request is responsible for calling the Content Harvester and passes to it the content requirements sent by the adaptive systems to start harvesting (or uploading) the appropriate content resources. On the other

[22] Documents in MongoDB are stored in collections. Collections are analogous to tables in relational databases.

hand, the slice retrieval request is responsible for calling the Slice Selector in order to retrieve a content slice according to the characteristics of the request sent by the adaptive system.

In order to organise the different types of requests received by the service, an additional component, named Request Coordinator was implemented. This component is responsible for identifying the type of the request and passing it to the relevant component. Additionally, the Request Coordinator is responsible for *blocking* any slice retrieval request before the CROCC service completes processing and indexing the incorporated content resources. The notion behind this blocking is that the coordinator prevents content slices from being retrieved for the adaptive system before all the potential slices that can meet the requirements of a request have been processed. This will make sure that the provided slice is the best match slice of the incorporated content resources.

5.5 Chapter Summary

This chapter presented a novel content provisioning service named CROCC. CROCC is a service which harvests content resources from closed and open corpora in their native form, builds a structure from each content resource based on its conceptual representation, and delivers content slices according to the requirements of individual adaptive systems. The chapter started by stating the influences derived from the state of the art review and how they impacted the core properties of the proposed service. The chapter then presented the design aspects of the CROCC service along with an explanation of how each component in the service influences the content provision process. The chapter also discussed how the service adheres to the key principles derived from the state of the art influences. After that, the chapter presented a prototype implementation of the service that has been carried out for the purpose of this research.

Finally, it is noteworthy to point out that a description of a preliminary version of the CROCC service was published in the following paper:

Bayomi, M. "*A Framework to Provide Customized Reuse of Open Corpus Content for Adaptive Systems*." In the Proceedings of the 26th ACM Conference on Hypertext & Social Media. HT ’15. Northern Cyprus, pp 315–318. ACM, 2015.

6. Evaluation of the CROCC Service

The previous chapter proposed a content-supply service named CROCC that aims to utilise the structure produced by the hierarchical text segmentation process in order to overcome the limitations of state of the art content-supply approaches. CROCC uses the C-HTS algorithm for text segmentation. Chapter 4 presented four experiments to evaluate different aspects of C-HTS. The **first** experiment was carried out in order to evaluate the performance of C-HTS in the hierarchical text segmentation task using two different datasets (section 4.5). The results showed that the performance of C-HTS is superior to the state of the art approaches. The **second** experiment was carried out in order to validate the efficacy of using Wikipedia as the underlying knowledge base for the semantic representation of text in C-HTS (section 4.7.1). The results showed that using Wikipedia as the knowledge base for C-HTS delivers better performance than using WordNet, even when using different relatedness measures with WordNet. The **third** experiment was carried out in order to validate the efficacy of using the semantic representation of text rather than its lexical representation (section 4.7.2). The results showed that using semantic representation in C-HTS outperforms lexical representation, even when using different similarity measures. The **fourth** experiment was carried out in order to evaluate the influence of the size of the knowledge base that C-HTS uses for semantic representation (section 4.8). The results showed that increasing the amount of knowledge in the knowledge base leads, on average, to improvements in C-HTS performance. While these experiments demonstrated the segmentation performance of C-HTS, this chapter aims to evaluate the extent to which the CROCC service, through the use of C-HTS, can enhance the discovery and reuse of content for adaptive systems.

As discussed in Chapter 5, CROCC is designed to supply content slices to adaptive systems according to their needs. In order to assess the performance of CROCC with regard to the supply of content slices, this chapter proposes a task-based experiment carried out in order to evaluate the quality of slices produced by CROCC. The chapter aims to address the fourth objective of this thesis (RO 4) (section 1.2).

6.1 Evaluation Methodology

The CROCC service is designed to provide content slices to adaptive systems according to their requirements. As mentioned in section 1.1, designing and building an adaptive system is not within the scope of this research. Therefore, the experiment presented in

this chapter did not focus on evaluating the process of content use within an actual adaptive system. Rather, the experiment focused on evaluating the content-supply mechanism of CROCC and the quality of the slices produced by the service, according to the specific requirements of a set of content requests that could be sent by an adaptive system.

The assumption is that, in order for content resources (from open and closed corpora) to be properly discovered and reused within different adaptive systems, the quality of the individual slices delivered must be guaranteed to adaptive system users. As a result, the approach chosen for this evaluation was to present a group of users with content slices produced by CROCC, where each slice was generated according to the specific requirements of a content request. Additionally, for each content request used in this experiment, another slice was generated by a baseline system to compare its quality against the slice generated by CROCC for the same request. The intention of this evaluation methodology is to focus on the evaluation of the content-supply mechanism proposed by CROCC in isolation from the adaptive functionality (models, authoring, etc.) of a specific adaptive system, as this could influence user perception of content quality.

As discussed in Chapter 5, CROCC is a content and adaptive system agnostic supply service. This means that the service can operate on any type of content regardless of its domain or structure and can supply content slices to any adaptive system regardless of its application area (e.g. Educational Adaptive System, Adaptive News System, etc.). In the evaluation carried out in this chapter, the application area of educational systems was chosen. Additionally, Computer Science was chosen as the domain. Amongst the different subjects in the computer science domain, Information Retrieval (IR) was selected as the focus. This decision was taken for a number of reasons. Firstly, an appropriate post-graduate course[1] in the field of Information Retrieval is taught in Trinity College Dublin by a Subject-Matter Expert (SME) who is working in the research group of which I am a member. Hence, the process of choosing the (closed corpus) content resources and the topics used in this experiment was guided by this SME. Secondly, there were a number of computer science researchers available within the ADAPT Centre[2] who could evaluate the quality of slices produced by CROCC and the baseline.

[1] https://www.scss.tcd.ie/modules/?m=CS7IS3
[2] https://www.adaptcentre.ie/

Content resources were acquired from closed and open corpora in the area of Information Retrieval, and were indexed by both CROCC and the baseline system. A total of 24 requests were submitted to each system, where each request has two main elements: a query and the number of sentences desired in the generated slice. The query element represents a simulation of an adaptive system's query. This query can include a specific topic or a short sentence. This query could be defined by the user of the adaptive system and forwarded to the content-supply system, or defined by the adaptive system based on the target user's model. The number of sentences element of the request represents a simulation of the level of granularity of the content slice required by the adaptive system. This level of granularity could be based on the preferences of the target user (e.g. reading speed or available time). After producing the slices, the two elements of each request, along with the two corresponding slices (one for each system) were presented to users to evaluate through a web application that was built for this experiment. Users were asked to evaluate the quality of each slice with regard to fulfilling the requirements of the request the slice was generated for. Evaluations submitted by users were then analysed, and results were derived.

This approach provides an evaluation of how CROCC enhances content discoverability and reusability for adaptive systems. For content discoverability, the assumption is that, if a slice generated by CROCC is highly preferred by the participant users, this means that the slice fulfils the requirements of the request and, hence, is properly discovered. Content reusability is assessed through this approach in two ways. First, the content used in this evaluation was generated from open and closed resources, neither of which were authored or designed specifically for this experiment. The content is also used in a range of different granularities. Thus, the original content, and newly sliced resources are being reused in a scenario that was not intended by the author of each individual content resource. Second, since content slices supplied to adaptive systems are ultimately presented to people, evaluating their quality using human assessors would best assess the effectiveness of the CROCC service in generating resources which are amenable for reuse. Each assessor can be seen as a user of an independent adaptive system. If a slice is highly preferred by multiple users, this means that it is a resource which is suitable for reuse by adaptive systems in the context specified by the information request.

6.2 Data and Content Sourcing

As discussed in Chapter 5, CROCC is designed to operate on content resources from closed and open corpus. This section discusses how the two datasets used in the experiment were acquired from open and closed corpora with regards to the Information Retrieval subject area. It is noteworthy to point out that this research focuses solely upon content resources available in English, as a first step and reserves multilingual content-supply for future work (section 7.3.1).

6.2.1 Closed Corpus Content Resources

The first dataset used in the experiment was built from a closed corpus of content resources. As mentioned in section 2.4.4.1, closed corpus content is (usually) provided by a domain expert. Hence, a book on the subject of Information Retrieval was provided by the course SME in PDF format. The book is "*Introduction to information retrieval*"[3] (Manning et al., 2008). This book is used by the SME in outlining the topics included in the course and is one of the recommended readings advised by the SME. The PDF file of the book consists of 21 chapters and has 581 pages.

In order to extract content from the PDF file of the book, the file was processed by the PDFX system (section 5.4.3) and an XML file was produced. The XML file contains the logical structure of the file where each section is encapsulated in a "*section*" XML tag and each content item (e.g. text segment, table, equation, figure, etc.) within each section is encapsulated in its corresponding XML tag. Figure 6.1 shows a sample of the XML structure produced by the PDFX system. Since the CROCC service is targeting text content only, the tags in the XML structure were used to remove the unneeded content items, such as tables and figures. Additionally, as the PDF file of the book contains sections that are not suitable for the content provision process (e.g. book preface, table of contents, exercises[4], bibliography, etc.), the XML tags corresponding to these sections were used to discard them. After that, the remaining content in each "*section*" tag was extracted and saved as a plain text file. The total number of content resources in this dataset after the pruning phase is 161 resources[5].

[3] http://informationretrieval.org

[4] As discussed in section 5.2.3, automatic pedagogical annotation algorithms could be applied to annotate content in these sections to use them in the content-supply process. However, such annotation information are reserved for the future work (section 7.3.3)

[5] Note that, a section can span over two or more pages. Hence, the final number of the extracted plain text files is not a portion of the total number of pages in the book (581 pages).

```
▼<outsider class="DoCO:TextBox" type="header" id="1">
   DRAFT! © April 1, 2009 Cambridge University Press. Feedback welcome.
  </outsider>
  <outsider class="DoCO:TextBox" type="header" id="2">1</outsider>
▼<region class="DoCO:TextChunk" id="3" confidence="possible">
   The meaning of the term information retrieval can be very broad. Just gett
   form of information retrieval. However, as an academic field of study, INF
   retrieval (IR) is finding material (usually documents) of an unstructured
   collections (usually stored on computers). As defined in this way, informa
   librarians, paralegals, and similar professional searchers. Now the world
   every day when they use a web search engine or search their email. 1 Infor
   overtaking traditional database- style searching (the sort that is going o
   give me your Order ID"). IR can also cover other kinds of data and informa
   "unstructured data" refers to data which does not have clear, semantically
   canonical example of which is a relational database, of the sort companies
   almost no data are truly "unstructured". This is definitely true of all te
   accepting that the intended notion of structure is overt structure, most t
   represented in documents by explicit markup (such as the coding underlying
   retrieval"; the term "search" is quite ambiguous, but in context we use th
  </region>
▼<region class="DoCO:FigureBox" id="F1.1">
   <caption class="deo:Caption" id="21" page="4" column="1">Figure 1.1</caption>
  </region>
```

Figure 6.1 A sample of the XML structure produced by the PDFX system

6.2.2 Open Corpus Content Resources

The second dataset used in the experiment was built from content resources harvested from open corpus. In order to harvest content resources from open corpus sources, the Content Harvester module implemented for this research is used. As mentioned in section 5.4.2, the open corpus content harvester is a wrapper component around the Bing Custom Search API. Therefore, a list of 20 queries was submitted to the Bing API through this wrapper. This list of queries was provided by the SME and each query represents a topic covered in the course. Examples of queries are: "*Boolean Retrieval*", "*Stemming*" and "*Link Analysis, Hubs and Authorities*". Each query was submitted to the search API along with the content repository where the data should be harvested from. The English Wikipedia encyclopaedia was selected as the target repository for the harvesting process. The reason for this is that Wikipedia is the largest encyclopaedia in existence which contains articles that cover a wide range of topics[6]. Additionally, Wikipedia is a collaborative effort that combines the knowledge of hundreds of thousands of people. Hence, articles in Wikipedia would be a convenient source for rich content resources.

For each submitted query, the top 10 results returned form the search API were downloaded as HTML files. Each document was then pruned by the open corpus Content

[6] https://en.wikipedia.org/wiki/Wikipedia:Size_comparisons

Pruner implemented for this experiment (section 5.4.3) and was saved as a plain text file. The total number of unique[7] content resources in this dataset is 147 resources.

6.3 Baseline System

In order to evaluate the efficiency of the CROCC service, it was necessary to compare its performance against other content-supply systems. However, as mentioned in section 2.5.2.2, state of the art approaches mainly rely upon the structure posed by a content resource and they can only produce coarse-grained content slices. To the best of the author's knowledge, and at the time of writing, there was no available content-supply system that can accept the number of sentences as a parameter to specify the granularity level of the produced content. Therefore, the existing state of the art content-supply systems would provide an artificially weak baseline against which to compare CROCC.

For this reason, a baseline system was developed to compare its performance against the CROCC service. The baseline system is based upon the Apache Lucene information retrieval software[8] (Białecki et al., 2012). Lucene is an open source full-text search library written in Java. Lucene provides an API that supports the performance of common search and search related tasks like indexing, querying, highlighting, language analysis and many others. There are two main reasons that Lucene was selected as the baseline system. Firstly, Lucene is a well-known, fast open source searching framework that utilises powerful, accurate and efficient search algorithms and is widely used in many projects (Mathew, 2018; Azzopardi et al., 2017; Hassen and Amel, 2017). Secondly, Lucene indexes content, in an inverted index, based upon the lexical representation of each document (terms in the document). Therefore, using it as a baseline in this experiment would implicitly provide an evaluation of the conceptual representation of content resources (proposed by its opponent, CROCC) versus lexical representation.

Lucene provides search over a collection of documents; where a document is essentially a collection of fields (e.g. title, body, date etc.). Documents are the unit of indexing and search in Lucene. Each field in each document can only store one kind of data, either binary, numeric, or text data. Lucene does not have a restriction over the structure of the fields to be indexed. It allows the indexing of any number of different fields that may vary from document to another. Lucene provides a search API that takes a search query

[7] Any duplicate document that results from the harvestin process was deleted
[8] http://lucene.apache.org/

and a set of fields to search within and returns a set of *documents* ranked by their relevance score to the query. However, as discussed in section 6.1, the experiment conducted in this chapter required the participant systems to accept, along with the query, the number of sentences as a parameter and retrieve a text slice according to that number. For this reason, a wrapper component around the Lucene framework was developed in order to conform with these experimental requirements. The following subsections describe how this wrapper component is employed to index content resources and to retrieve content slices.

6.3.1 Document Indexing

In order to index content resources within Lucene in a manner that allows the generation of content slices according to granularity level (number of sentences), the wrapper component started by dividing each content resource (the plain text files produced as described in section 6.2) into sentences. After that, each sentence was indexed as an independent *Document* object in Lucene along with the id of the document the sentence belongs to. The text of each sentence was analysed using the *English Analyser* of Lucene. The analyser started by breaking each sentence into small indexing elements – tokens. Stopwords[9] were then removed and the remaining tokens were stemmed (using Porter stemmer) and indexed in the inverted index created by Lucene. Figure 6.2 illustrates how a document is indexed in Lucene.

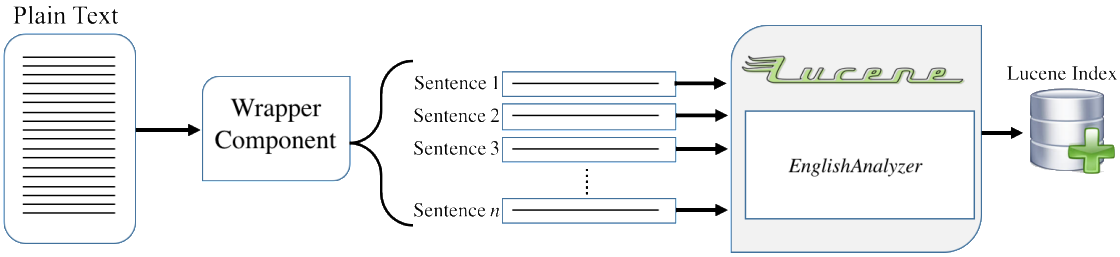


Figure 6.2 An illustration of how a document is indexed in Lucene

6.3.2 Slice Generation

To generate a slice based on the elements of the request (query and number of sentences), the wrapper component works as follows:

- When a request is received, the wrapper starts to search the Lucene index using the text of the query. It utilises the Lucene retrieval mechanism that matches the

[9] This analyser uses the default stopwords list that is bundled with Lucene.

query terms with the terms in the inverted index. After that, sentences that match the query are returned along with their relevance score.

- For sentences that do not match the query, their relevance score is set to zero.
- Using the number of sentences element of the request, say *x*, the wrapper iterates over all the returned sentences and calculates the accumulated score of every *x* adjacent sentences that belong to the same document.
- After that, sentences that have the best accumulated score are returned as a text slice.

This mechanism guarantees that the returned slice: (1) consists of sentences that best match the query and (2) conforms with the "number of sentences" parameter of the request. The flowchart presented in Figure 6.3 illustrates the process of generating a text slice by the baseline system according to the request elements.

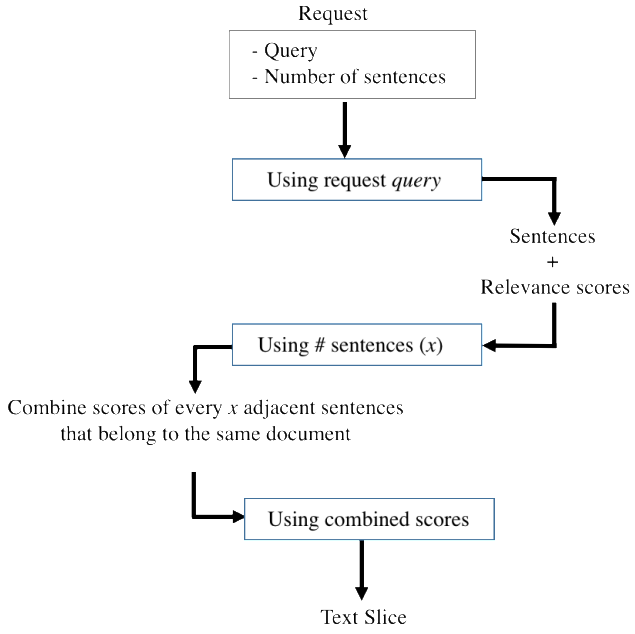


Figure 6.3 A flowchart of the slice generation process by the baseline system

6.4 Experimental Setup

6.4.1 Concept Vector Cut-off Parameter

As discussed in section 5.2.4, CROCC indexes content slices based on the concepts assoicated with each slice. As mentioned in section 4.3.1, each slice is mapped to a weighted centroid vector of concepts that is built based on ranking all Wikipedia concepts by their relevance weight to the slice. With concept weights being zero for most of the Wikipedia concepts (as no term in the slice is associated with these concepts), this centroid vector is very sparse. Nevertheless, given that each term in the slice to be indexed

may still be related to a large number of concepts (> one thousand concepts), indexing the entire list of related concepts for every slice would incur significant storage and computation costs, and is therefore not feasible. Therefore, only the concepts with the highest relevance weights should be considered. Hence, a vector cut-off parameter is set in order to reduce index size.

We ran the experiment presented in section 4.5 using different sizes of the centroid vector to evaluate the impact of its size on C-HTS performance. We ran the experiment using all concepts in the centroid vector of each text block, the top 50 concepts and the top 100 concepts. The results showed that there is no significant difference in C-HTS performance and thus, using the top 50 concepts that have the highest relevance weights produces the same structure but with less computation cost. Additionally, in (Egozi et al., 2011), the authors proposed a concept-based indexing and retrieval approach based on Explicit Semantic Analysis (ESA) where documents were indexed based upon their conceptual representation from a concept space built from Wikipedia. In this work, in the indexing process, setting the cut-off parameter to 50 gave the best results in their retrieval task[10]. Thus, based on these findings, in experiment conducted in this chapter, the cut-off parameter for the concept vector was set to 50 concepts.

As discussed in section 5.2.6, the *request analyser* maps the query sent by the adaptive system to a weighted centroid vector of concepts. Since the query is derived from a much shorter text fragment and contains few terms, the value for the cut-off parameter is not necessary to be the same as that used in the indexing process. Hence, in this experiment, the cut-off parameter was not set for the centroid vector of the received query and all concepts in that vector were used by the *slice retriever* sub-module of the Slice Selector.

6.4.2 Datasets Indexing and Slices Generation

The two datasets built for this experiment were indexed in both systems. For the CROCC service, a hierarchical structure was built for each content resource using the Structure Builder (section 5.4.4). After that, the cut-off parameter was set for the Slice Indexer (section 5.2.4), as discussed in the previous section, and the produced slices were then indexed in the Content Repository (section 5.2.5). For the baseline system, the content resources of both datasets were indexed in a Lucene index as described in section 6.3.1.

[10] They have also experimented with indexing the 100 most relevant concepts instead of top 50, and found no significant impact on the performance.

It is noteworthy to mention that both datasets were indexed in a single index in each system. The reason for this is to assess the assumption that CROCC is a content agnostic service.

After indexing the two datasets, the systems were ready to receive content requests. To build content requests, there was a need to specify the two main elements of each request: the query and number of sentences. Six topics in the IR subject area were selected to be used as the query element of each submitted request. The queries generated from these topics are:

- Query 1: *Boolean Retrieval*
- Query 2: *Inverted Index*
- Query 3: *Stemming and Lemmatisation*
- Query 4: *TF-IDF*
- Query 5: *Relevance feedback*
- Query 6: *Precision, Recall and F-score*

These topics were provided by the course SME in which they were intended to be typical of a student's information needs based on the topics covered in the course. For each of these topics, four different requests were built where each request had a different value for its "number of sentences" element. The values defined for this element were: 4, 5, 6 and 7 sentences. Thus, a total of 24 content requests were built (4 different sizes for each of the 6 topics) and were submitted to both system.

For the CROCC service, requests were received by the Slice Selector module of the service (section 5.2.6). As discussed in section 6.4.1, the cut-off parameter was not set for the request analyser module while building the centroid vector of concepts from the query. Using this concepts vector and the number of sentences, slices that best match each individual request were generated and returned. For the baseline system, the requests were received by the wrapper component and slices that best match each individual request were generated and returned as described in section 6.3.2. A total of 48 slices were returned (2 for each request) and saved in a database to be provided to participants for evaluation (see next section). The slices returned by both systems were generated from the closed and open corpus content resources harvested for this experiment (section 6.2). Figure 6.4 depicts the distribution of slices returned by each system. Table 6.1 shows a sample of slices generated by both systems.

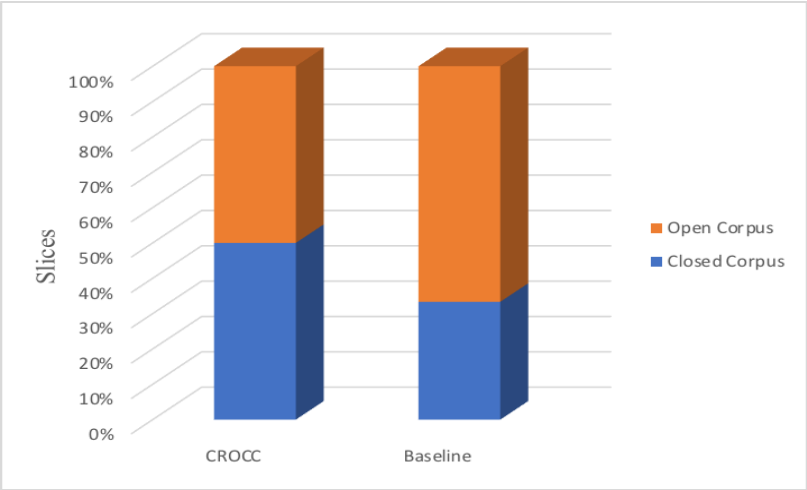


Figure 6.4 Slices distribution over closed and open corpora

Table 6.1 A sample of slices generated by both systems

Query and Size	CROCC	Baseline
Boolean Retrieval 4 sentences	1. A general problem with Boolean search is that using AND operators tends to produce high precision but low recall searches, while using OR operators gives low precision but high recall searches, and it is difficult or impossible to find a satisfactory middle ground 2. In this chapter, we have looked at the structure and construction of a basic inverted index, comprising a dictionary and postings lists 3. We introduced the Boolean retrieval model, and examined how to do efficient retrieval via linear time merges and simple query optimization 4. In Chapters 2–7 we will consider in detail richer query models and the sort of augmented index structures that are needed to handle them efficiently	1. Worked example Commercial Boolean searching: Westlaw.westlaw 2. Westlaw (http://www.westlaw.com/) is the largest commercial legal search service (in terms of the number of paying subscribers), with over half a million subscribers performing millions of searches a day over tens of terabytes of text data 3. The service was started in 1975 4. In 2005, Boolean search (called ``Terms and Connectors'' by Westlaw) was still the default, and used by a large percentage of users, although ranked free text querying (called ``Natural Language'' by Westlaw) was added in 1992

6.4.3 Evaluation System

To conduct the comparative evaluation between the two systems, a web application was built. The slices produced from both systems were divided into four groups; each group had six pairs of slices (a slice generated from each system) corresponding to each of the search topics and to one of the defined sizes (4, 5, 6 or 7). To make sure that all slices get the same amount of evaluations, slices were distributed throughout the groups according to their sizes as listed in Table 6.2.

Table 6.2 Slice sizes for each topic in each group

	Group 1	Group 2	Group 3	Group 4
Boolean Retrieval	4	5	6	7
Inverted Index	5	6	7	4
Stemming and Lemmatisation	6	7	4	5
TF-IDF	7	4	5	6
Relevance feedback	4	5	6	7
Precision, Recall and F-score	5	6	7	4

To make sure that all groups get assigned the same amount of participants, when the first user logged in to the evaluation system, s/he was randomly assigned a group. The next user was then randomly assigned a group from the remaining unassigned groups. This continued until all the groups were assigned users. The process was then repeated for the next set of users who logged in to the system. This ensured an even spread of assessment. For each query in the group, the query and the size of the slice were presented to the user at the top of the evaluation page followed by the two generated slices from both systems side-by-side. In order to guarantee that there is no incentive for users to be biased towards either of the systems, the users were not aware of which slice was produced using which system.

Each user in the experiment was asked to evaluate each slice according to the following characteristics:

1- **Relevance**: if the slice is relevant to the query.

2- **Informativeness**: if the slice contains all the necessary information compared to its size (e.g. 5 sentences).

3- **Cohesion & Readability**: if the slice is easy to read and the flow of reading is not broken.

4- **Overall**: the overall quality of the slice.

The users were asked to evaluate each characteristic on a six-point Likert scale (ranging from one to six, where one is the lowest quality and six is the highest). For the sake of data completeness, each user was asked to fill in answers for all the evaluation characteristics. A text box was provided in case the user had any comments regarding the quality of the slices or regarding the difference between them. The users were allowed to leave this box empty if they did not have any comments. After each slice was evaluated individually, the user was asked to indicate which slice they preferred. In most cases this

characteristic was automatically set by the system based upon the user's individual evaluation of the two slices, while still allowing for manual adjustment if the user so wished. However, if the user evaluated the two slices equally, then neither slice was preselected, and they had to manually make a selection. Figure 6.5 shows the evaluation screen of the system.

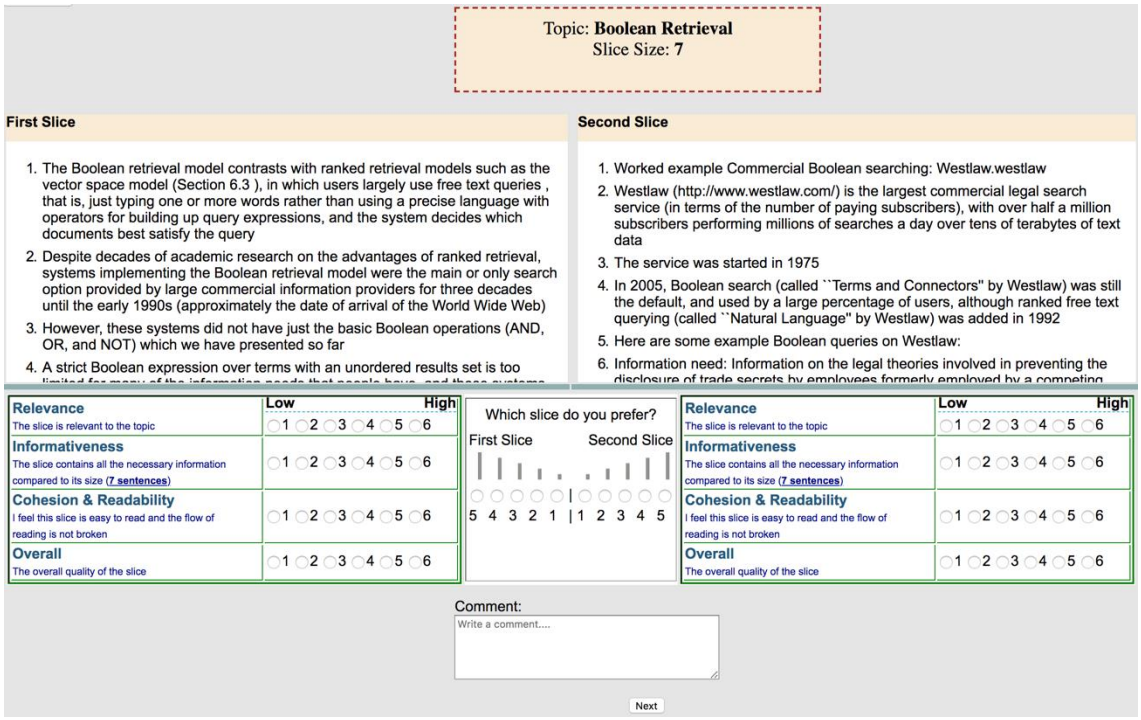


Figure 6.5 Evaluation System

6.5 Results

An open call for participation[11] in the experiment was made on the ADAPT Centre mailing lists and on the mailing lists of research groups whose members are interested in the IR field. Forty-eight users participated in the experiment where each slice was evaluated by more than ten users[12]. The final results were analysed with the following aspects in mind:

1- The general performance of each system.

2- The query element of the request.

3- The "number of sentences" element of the request.

[11] This experiment conforms to ethical research conduct and was approved by the Research Ethics Committee of the School of Computer Science and Statistics, Trinity College Dublin.

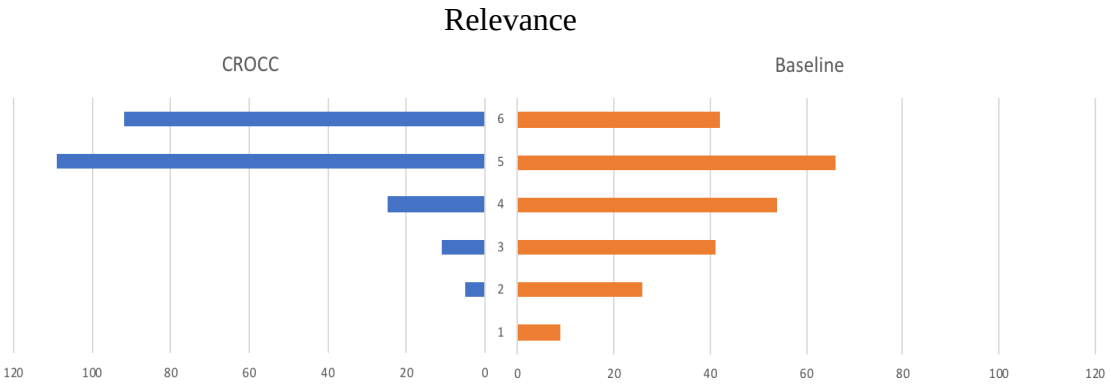
[12] Some users did not complete the experiment and exited before evaluating all six requests assigned to them. These users were removed from the system. As a result, a number of completed evaluations were randomly removed in order to balance the number of participants per slice. The final number of evaluations considered in the analysis presented here is 10 participants per slice.

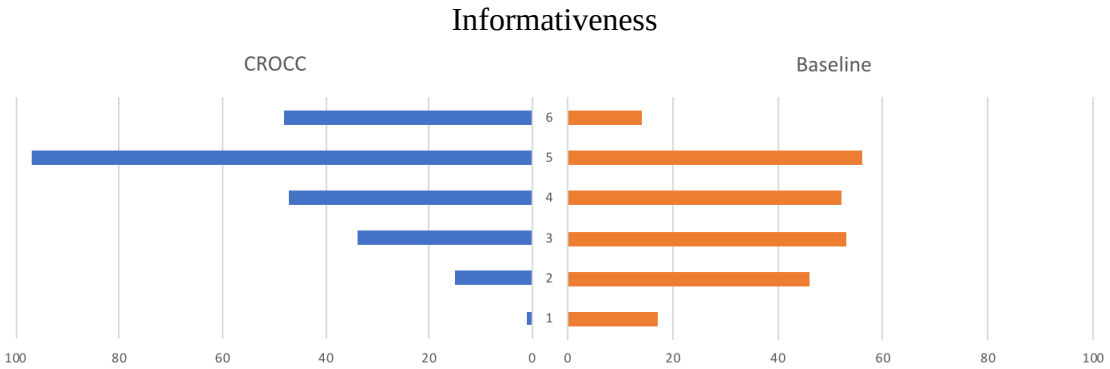
6.5.1 General Performance

In this analysis, the performance is assessed based on the quality of the 24 slices generated by each system. Table 6.3 reports the mean scores of user evaluations for each characteristic[13]. As can be observed, the quality of slices generated by CROCC is, on average, higher than the quality of slices generated by the baseline system in all criteria. Figure 6.6 depicts the distribution of user evaluations for each criteria for both systems. Moreover, the statistical results from a paired t-test have revealed that the difference between the quality of slices is significant with p-values less than 0.05 across all criteria[14].

Table 6.3 Mean scores of user evaluations for all slices produced by each system

System / Criteria	CROCC	Baseline
Relevance	5.104*	4.158
Informativeness	4.529*	3.596
Cohesion & Readability	4.412*	3.846
Overall	4.521*	3.671
Preference	1.688*	0.698





Cohesion & Redability

[13] Since writing a comment by the users was optional, there was no enough comments that can be reliable in the results analysis.

[14] In the tables, the asterisk symbol * denotes that the difference between the quality of slices is statistically significant with $p < 0.05$

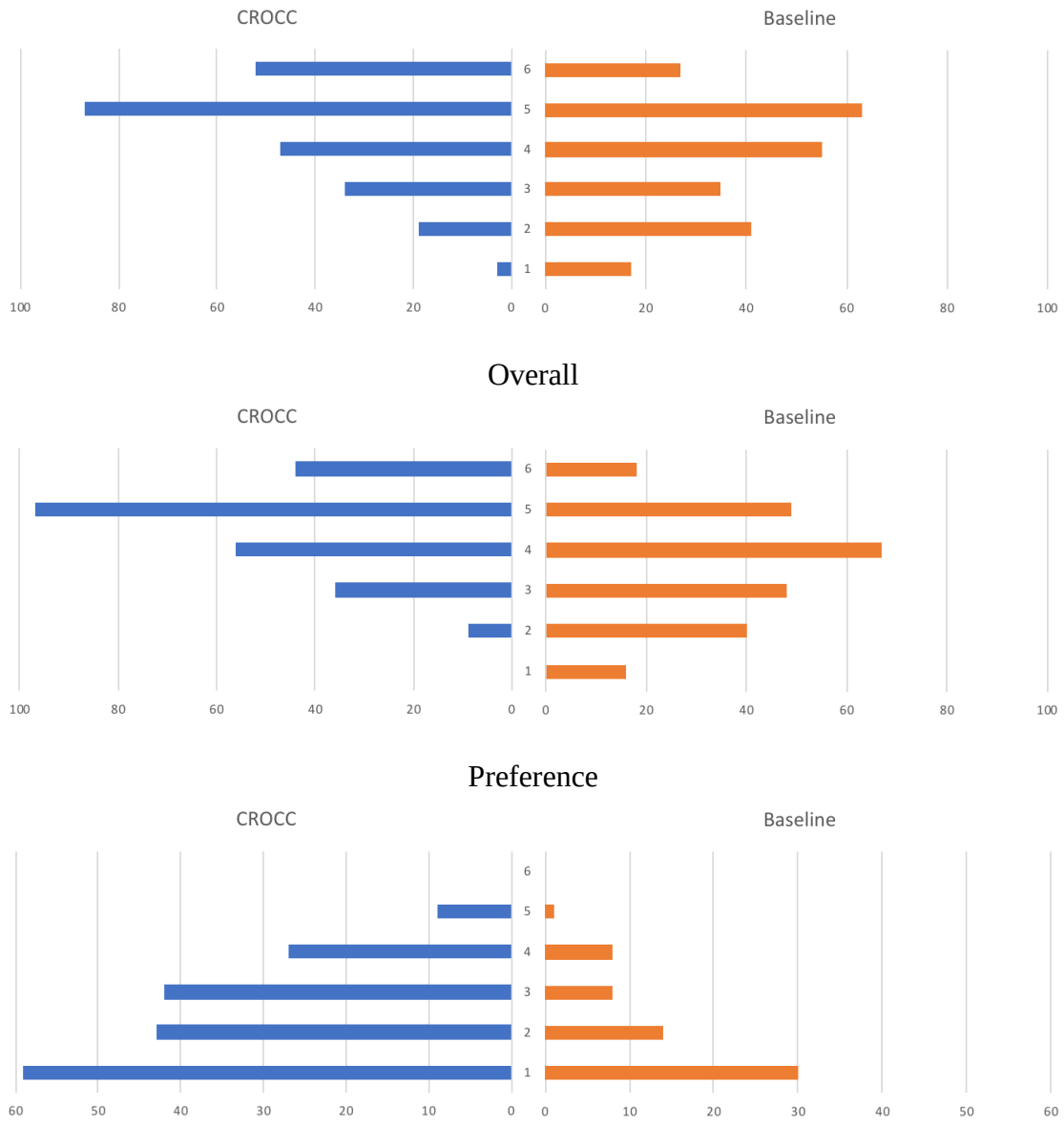


Figure 6.6 Distribution of general user evaluations for each criteria

6.5.2 The Query Element of the Request

In order to gain more insight into the performance of each system, the results were analysed based on the quality of slices produced by each system in accordance with the query element of the requests. Table 6.4 reports the mean scores of the user evaluations for slices produced for each query. As can be observed, the quality of slices generated by CROCC is, on average, higher than slices generated by the baseline system for each query. These results provide a positive indication of the performance of CROCC in relation to the ***discovery*** of content slices that best match the query of the request. Additionally, since CROCC uses the semantic representation of content (section 5.2.3) and the query (section 5.2.6), the results demonstrate that such semantic representation is better than the traditional lexical representation utilised by the baseline system.

Table 6.4 Mean scores of user evaluations for slices produced for each query by each system

Query \ Criteria		Relevance	Informa-tiveness	Cohesion & Readability	Overall	Preference
Query 1: Boolean Retrieval	CROCC	5*	4.375*	4.125*	4.4*	2.3*
	Baseline	3.225	2.725	3.175	2.75	0.3
Query 2: Inverted Index	CROCC	5.15	4.625	4.45	4.55	1.425
	Baseline	4.8	4.35	4.125	4.225	0.875
Query 3: Stemming and Lemmatisation	CROCC	5.275	4.65	4.775	4.625	1.35
	Baseline	5	4.275	4.725	4.45	1.025
Query 4: TF-IDF	CROCC	5.325*	4.675*	4.35*	4.6*	2.45*
	Baseline	3.65	2.9	3.025	3.075	0.075
Query 5: Relevance feedback	CROCC	5.303	4.636	4.758	4.758	0.758
	Baseline	5.090	4.515	4.667	4.636	0.545
Query 6: Precision, Recall and F-score	CROCC	4.825*	4.275*	4.275*	4.375*	1.975*
	Baseline	3.275	2.8	3.35	2.9	0.3

6.5.3 Number of Sentences Element of the Request

The purpose of this analysis is to investigate how each system is able to generate a content slice, from the incorporated content resources, according to the "number of sentences" (level of granularity) element of each individual request. Table 6.5 reports the mean scores of the user evaluations for slices produced by each system. As can be observed, the quality of slices generated by CROCC is, on average, higher than the quality of slices generated by the baseline system with regards to all sizes in all criteria.

The results, hence, prove that the structure built for each individual content resource by CROCC enables the service to generate slices that best match the different levels of granularity in each individual request. This in turn proves that content resources processed by CROCC are amenable for reuse in different adaptive systems.

Table 6.5 Mean scores of user evaluations for slices with regards to number of sentences

Slice Size \ Criteria		Relevance	Informa-tiveness	Cohesion & Readability	Overall	Preference
Size = 4	CROCC	4.933*	4.133*	4.117	4.267*	1.667*
	Baseline	3.95	3.45	3.8	3.533	0.833
Size = 5	CROCC	4.983*	4.5*	4.567*	4.433*	1.567*
	Baseline	4.233	3.7	3.95	3.767	0.517
Size = 6	CROCC	5.1*	4.583*	4.233*	4.5*	1.55*
	Baseline	4.183	3.633	3.817	3.733	0.6
Size = 7	CROCC	5.4*	4.9*	4.733*	4.883*	1.967*
	Baseline	4.267	3.6	3.817	3.65	0.483

6.5.4 Discussion

This experiment proposed an evaluation for the performance of the CROCC service as a content-supply service. The experiment focused on evaluating the quality of the slices produced by CROCC according to the elements of each individual request. Comparing the quality of slices produced by CROCC against slices produced by the baseline system demonstrated that CROCC outperformed the baseline system in all characteristics specified in the experiment.

According to the assumption posed in section 6.1, which is: *in order for content resources (from open and closed corpus) to be properly discovered and reused within different adaptive systems, the quality of individual slices delivered must be guaranteed to adaptive system users*, these results, therefore, clearly prove that CROCC is capable of producing content slices **from open and closed corpus** resources that fulfil the requirements of the content requests. This in turn means that the slices produced by CROCC are properly **discovered**. Additionally, as slices produced by CROCC are highly preferred by the participant users, this means that CROCC is capable of producing content slices which are suitable for **reuse** by adaptive systems in the context specified by individual content requests.

6.6 Chapter Summary

This chapter presented a task-based experiment that aimed to address the fourth objective of this thesis; namely to evaluate the extent to which the CROCC service can enhance the discovery and reuse of content for adaptive systems. In particular, the experiment aimed to evaluate the quality of the content slices produced by CROCC. The chapter described the methodology that has been followed in the evaluation process and the baseline system that has been developed to compare against the CROCC service. The chapter described how content resources from closed and open corpora were acquired along with a description of how these resources were processed and indexed in each system used in the experiment. The evaluation system was also described along with the steps that participants have followed to complete the experiment. Finally, the chapter presented the analyses carried out on the evaluation results, along with the findings derived from this analysis.

7. Conclusion and future work

This chapter presents the final conclusions for the research presented in this thesis. The chapter reviews the research questions and objectives presented in Chapter 1 of this thesis and discusses the extent to which these objectives have been achieved, and the research question answered. It also revisits the overall contributions of this thesis. Finally, the chapter outlines possible future research directions on the basis of the research outcomes achieved in this thesis.

7.1 Research Question, Objectives and Achievements

The research question which this thesis initially sought to answer was:

To what extent can the semantic representation of unstructured textual content be exploited by novel text segmentation approaches to build a document structure?

Also, to assess whether the structure produced by the proposed approaches is of benefit for adaptive systems, a further question was posed:

Can the produced structure be used to enhance the discoverability and reusability of content for adaptive systems?

In order to address these research questions, four research objectives were outlined (section 1.2.1). The following sub-sections discuss each objective and how each objective was achieved in this thesis.

7.1.1 Research Objective 1

The first research objective is: "*Perform a state of the art survey on NLP techniques, specifically text segmentation as a technique for structuring textual content. The aim of this survey is to investigate how text segmentation is used to analyse and understand text to produce a structure from unstructured textual documents. Additionally, perform a state of the art survey on adaptive systems as content adaptation applications, to investigate how they process content and the different techniques they utilise in order to facilitate the discovery and reuse of this content. The survey should also review how state of the art adaptive systems utilise NLP techniques in order to provide adaptive content".*

To achieve this objective, an integral part of this thesis involved conducting a state of the art review of NLP techniques and a focused review on text segmentation. The review

classified different approaches to text segmentation and highlighted different categorisation criteria of this task. The review also focused on hierarchical text segmentation and investigated how hierarchical text segmentation is used to analyse text to produce a structure from unstructured textual documents. The review showed that state of the art approaches are limited by the fact that they can only process the information that they can 'see'. In other words, they are based on the lexical and/or syntactic representation of text. However, a representation based solely on the endogenous knowledge in the documents themselves does not reveal much about the meaning of the text. Building on the influences derived from this review, two novel approaches to hierarchical text segmentation were presented in Chapter 3 and Chapter 4. Both approaches utilise external knowledge resources in order to enrich text and infer more information about text constituents.

The field of NLP and specifically hierarchical text segmentation has mainly focused on techniques that can be used to process text documents, but not, however, on how these techniques can be utilised to produce tailored content, adapted to the needs of individual users. Conversely, research in the field of Adaptive Systems has primarily focused on methods and techniques of delivering adaptive content to individual users. Hence, a comprehensive review on adaptive systems has been conducted. The purpose of this review was to investigate how adaptive systems process content to facilitate its discoverability and reusability. The review discussed the anatomy of adaptive systems, their models and in particular their content model. In order to better illustrate how adaptive systems process different types of content, closed and open corpus content models were reviewed. The review also presented the different techniques utilised by adaptive systems to discover content resources that best match their users' needs. Furthermore, current NLP techniques used by adaptive systems were reviewed to gain more insight into how adaptive systems utilise these techniques in processing content resources and how these techniques contribute to the provision of adaptive experiences to users. Building on the influences derived from this review, a new content-supply service was designed, which is presented in Chapter 5.

7.1.2 Research Objective 2

The second research objective is: "*Examine the different methods and techniques that can be used to enhance the performance of text segmentation using the semantic representation of text, and develop a new text segmentation approach to enhance the understanding*

of unstructured textual documents. This also involves the evaluation of the effectiveness of the proposed approach in processing and structuring content".

To achieve this objective, a new hierarchical text segmentation approach named OntoSeg (**Onto**logical **Seg**mentation) was proposed (Chapter 3). The aim of OntoSeg was to understand the semantic meaning behind text in order to build a conceptual structure. In contrast to state of the art approaches, OntoSeg replaces the traditional lexical representation of text with concepts extracted from an ontology. OntoSeg starts by extracting named entities from the target text. Each entity is then mapped to its class or classes in the DBpedia ontology. After that each sentence is represented as a vector of entities where each element in that vector is represented as a set of classes. To identify how each sentence is similar to its adjacent sentences, two different similarity measures were calculated, semantic similarity and lexical similarity. Using these similarity measures, OntoSeg applies a Hierarchical Agglomerative Clustering (HAC) approach to iteratively cluster text segments that are deemed to be similar to each other and produce a tree-like hierarchy of the text.

To evaluate the effectiveness of OntoSeg in processing and structuring content, a set of experiments have been conducted. Experimental results showed that, although OntoSeg is able to produce a hierarchical structure out of text based on its semantic representation, it did not perform well against the state of the art approaches.

As a result, a new hierarchical text segmentation approach, named C-HTS, was proposed (Chapter 4). C-HTS (**C**oncept-based **H**ierarchical **T**ext **S**egmentation) relies on the semantic *relatedness* between text constituents rather than the semantic *similarity* used in OntoSeg. C-HTS relies on the *explicit* semantic representation of text, a method that replaces keyword-based text representation with concept-based features, automatically extracted from massive human knowledge repositories such as Wikipedia. C-HTS represents the meaning of a piece of text as a weighted vector of knowledge concepts, in order to reason about text. Relatedness between the atomic units of text is measured using this semantic representation and a Hierarchical Agglomerative Clustering algorithm is then applied to grow coherent segments of the text and a tree-like hierarchy of the text is produced.

The performance of C-HTS was compared against the state of the art hierarchical text segmentation approaches, using two datasets that are designed specifically for the evaluation of such systems. The results showed that C-HTS outperformed the state of the art

and also outperformed OntoSeg. Additionally, in order to validate the efficacy of using Wikipedia as the underlying knowledge base for conceptual representation of text in C-HTS, an experiment was carried out where WordNet was used as the underlying knowledge base for C-HTS. The results demonstrated that using Wikipedia as the knowledge base for C-HTS delivers better performance than using WordNet, even when using different relatedness measures with WordNet. Another experiment was also carried out in order to validate the efficacy of using the semantic representation of text rather than its lexical representation. The results showed that using semantic representation of text in C-HTS outperforms the lexical representation approach even when using different lexical similarity measures. In order to evaluate the influence of the size of the knowledge base that C-HTS uses for semantic representation, an experiment was carried out where three different snapshots of Wikipedia over different years were used with C-HTS. The results showed that increasing the amount of knowledge in the knowledge base leads, on average, to improvements in C-HTS performance.

7.1.3 Research Objective 3

The third research objective is: "*This PhD research takes adaptive systems as the target application scenario. To enhance the content discoverability and reusability, it is important to understand the structure of that content. The proposed hierarchical text segmentation approach makes it possible to build a structure out of content resources based on the semantic representation of text. In this context, a new content-supply service that utilises the structure produced by the proposed segmentation approach needs to be built. The design of this service should be focused on exploiting the produced structure in order to overcome the limitations of the state of the art content-supply approaches*".

To achieve this objective, a novel content-supply service named *CROCC* (**C**ustomised **R**euse of **O**pen- and **C**losed-corpus **C**ontent) was proposed (Chapter 5). CROCC was built based on the influences derived from the state of the art review presented in Chapter 2. The aim of CROCC is to utilise the structure produced by the C-HTS algorithm in order to overcome the limitations of the state of the art content-supply approaches. CROCC is a service which harvests content resources from open and closed corpus in their native form and builds a structure out of each content resource without the reliance upon its original structure. The service builds the structure of a content resource based on its conceptual representation and delivers content slices according to the needs and requirements of individual adaptive systems.

CROCC was designed using a flexible architecture that allows for plugging-in, removing, enabling, or disabling alternative components or algorithms at runtime as well as design time of the service. The CROCC service was offered as an intelligent content provision framework, which comprises six modules. The first module is the Content Harvester which is responsible for the automatic acquisition of content resources based on the content requirements of the adaptive system. The second module is the Content Pruner. Because CROCC does not rely upon the original structure posed by each harvested content resource, the main task of the Content Pruner is to identify and remove the unnecessary fragments within content resources and convert each individual content resource into a plain text file. The third module is the Structure Builder. The main task of this module is to build a structure out of the harvested and pruned content resources based on their conceptual representation. This module utilises the C-HTS algorithm in order to complete this task. Recall from Chapter 4 that the output of C-HTS is a tree-like hierarchy for each individual content resource. Using this hierarchy, the fourth module in CROCC, called Slice Indexer, starts to index slice objects produced in each level of the hierarchy produced by the Structure Builder where each slice object contains a metadata description for each slice. These slice objects are indexed in the Concept Index of the Content Repository module (fifth module in CROCC). Besides the concept index, the Content Repository module contains the Text Index. This index is used to store the textual content of the harvested resources. This textual content is indexed as a list of sentences that is used later to generate the textual content of the selected slice. The last module in CROCC is the Slice Selector. This module is considered the interface of the CROCC service with the adaptive systems. The main task of this module is to receive the request sent by an adaptive system, process it, and retrieve the slice that best matches this request.

This design of the CROCC service has successfully demonstrated that the service adheres to the different key principles derived from the state of the art influences presented in section 5.3. In addition to this design, a prototype implementation of the CROCC service has been carried out in order to be used in the evaluation conducted in Chapter 6.

7.1.4 Research Objective 4

The fourth research objective is: "*Evaluate the extent to which the proposed content-supply service can enhance the discovery and reuse of content for adaptive systems*".

To achieve this objective, a task-based experiment has been carried out. The main purpose of this experiment was to evaluate the quality of the content slices produced by CROCC.

The experiment did not focus on evaluating the process of content use within an actual adaptive system. Rather, the experiment focused on evaluating the content-supply mechanism of CROCC and the quality of the slices produced by the service, according to the specific requirements of a set of content requests that could be sent by an adaptive system. The assumption is that, in order for content resources (from open and closed corpora) to be properly discovered and reused within different adaptive systems, the quality of the individual slices delivered must be guaranteed to adaptive system users. As a result, the approach chosen for this evaluation was therefore to present a group of users with content slices produced by CROCC where each slice was generated according to the specific requirements of a content request.

The application area chosen for this experiment was Educational Systems. Additionally, the field of Information Retrieval was selected as the subject area of the experiment. A baseline system was developed in order to compare its performance against the CROCC service. Content resources from closed and open corpora related to IR were harvested, processed and indexed by CROCC and the baseline system. A total of 24 content requests were constructed and for each request a slice was generated by each system. The produced slices (48 slices) were then segmented into four groups and an evaluation system was built in order to present these groups to participants for evaluation. An open call for participation in the experiment was made on a number of mailing lists and a total of forty-eight users participated in the experiment where each slice was evaluated by more than ten users.

In this experiment, each user was asked to evaluate each slice according to four different characteristics: Readability, Informativeness, Cohesion & Readability and Overall. Additionally, the user was asked to indicate which slice they preferred. The evaluation system also had a text box in case the participant had any comments regarding the quality of the slices or regarding the difference between them.

The evaluations submitted by participants were analysed and the results demonstrated that the quality of slices generated by CROCC is, on average, higher than the quality of slices generated by the baseline system in all criteria. This in turn demonstrated that slices produced by CROCC (from closed and open corpus resources) fulfil the requirements of the content requests and, hence, are properly discovered. Additionally, as slices produced

by CROCC are highly preferred by the participant users, this means that CROCC is capable of producing content slices which are suitable for reuse by adaptive systems in the context specified by individual content requests.

7.2 Contributions

This section revisits the contributions from the research of this thesis, which were presented in section 1.3. The research of this thesis has made three notable contributions: one major contribution and two minor contributions.

The major contribution of this thesis is the use on NLP techniques, particularly text segmentation, to produce a hierarchical structure from text documents and the use of this structure by a content-supply service to enhance content discoverability and reusability for adaptive systems. To build a structure from text documents, this thesis proposed two novel hierarchical text segmentation algorithms based on the semantic representation of content, OntoSeg and C-HTS. OntoSeg used the semantic similarity between text segments based on an ontology and uses a Hierarchical Agglomerative Clustering algorithm to build a hierarchical structure of text based on its semantic representation. Evaluation results demonstrated that although OntoSeg is able to produce a hierarchical structure of text based on its semantic representation, it did not perform well against the state of the art approaches. These findings indicated that the performance of OntoSeg can be improved through improved understandability of text, by exploring the semantic relatedness between text blocks rather than using the semantic similarity. As a result, the C-HTS algorithm was proposed. C-HTS used the explicit semantic representation of text to measure the semantic relatedness between text blocks. It represented the meaning of a piece of text as a weighted vector of knowledge concepts automatically extracted from the massive human knowledge repository, Wikipedia. Similar to OntoSeg, *C-HTS* produced the content of a single document as a tree-like hierarchy. Evaluation results have shown that C-HTS outperformed the state of the art approaches on two datasets that are designed specifically for the evaluation of hierarchical text segmentation. The results also demonstrated that using the semantic relatedness in C-HTS yielded a better hierarchical structure of text than using the semantic similarity employed by OntoSeg.

This thesis also presented a novel content-supply service named CROCC. CROCC is a service which harvests content resources from open and closed corpus in their native form and builds a structure out of each content resource without the reliance upon its original structure. CROCC utilises the C-HTS algorithm to build a structure of the harvested

content resources based on their semantic representation. Using this structure, the service delivers content slices that best match the needs and requirements of individual adaptive systems. The aim of CROCC is to enhace content discoverability and reusability for adaptive systems. This thesis also presented a task-based experiment to evaluate the extent to which the CROCC service can enhance the discovery and reuse of content for adaptive systems. The main focus of this experiment is to evaluate the quality of content slices produced by the CROCC service according to the specific requirements of a content request that might be sent by an adaptive system. The experiment focused on a specific application are and specific subject area. Content resources were collected from closed and open corpus in the specified subject area. A baseline system was developed in order to compare its performance against CROCC. Evaluation system was built to present content slices produced by each system to the participant users for evaluation. Experimental results demonstrated that the quality of slices produced by CROCC are highly preferred by users than slices produced by the baseline system.

A minor contribution of this thesis is the concept space that was built from Wikipedia for the purpose of this research. The concept space was built from a Wikipedia snapshot (April 2017) to be used for the explicit semantic analysis of text within C-HTS. This concept space is publicly available[15] and can be used by researchers who work on tasks related to explicit semantic analysis. Another minor contribution is the implementations of the two hierarchical text segmentation algorithms proposed in this thesis, OntoSeg and C-HTS. Implementations of both algorithms have been open-sourced and made publicly available[16,17].

The contributions of this research have also resulted in the following academic publications:

- Bayomi, M., Levacher K., Ghorab, M.R., and Lawless, S. "*OntoSeg: a Novel Approach to Text Segmentation using Ontological Similarity*". In the proceedings of the 5th ICDM Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction, ICDM SENTIRE. Held in conjunction with the

[15] https://goo.gl/JZhEvm
[16] https://github.com/bayomim/OntoSeg
[17] https://github.com/bayomim/C-HTS

IEEE International Conference on Data Mining, ICDM 2015. Nov 14th, 2015. Atlantic City, NJ, USA.

- Bayomi, M. & Lawless, S. "*C-HTS: A Concept-based Hierarchical Text Segmentation approach*". In the Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). Miyazaki, Japan: European Language Resources Association (ELRA).
- Bayomi, M. "*A Framework to Provide Customized Reuse of Open Corpus Content for Adaptive Systems*." In the Proceedings of the 26th ACM Conference on Hypertext & Social Media. HT '15. Northern Cyprus, pp 315–318. ACM, 2015.

Additionally, a publication describing the CROCC service and its evaluation (detailed in Chapter 5 and Chapter 6) is underway and will target the ACM Hypertext conference.

7.3 Further Work

This section discusses potential further work that could be undertaken for the research in this thesis.

7.3.1 Multilingual Content-Supply

CROCC has primarily focused on harvesting and processing content resources that are written in English. The main module that processes the textual content of the harvested resources is the Structure Builder module (section 5.2.3) that utilises the C-HTS algorithm presented in Chapter 4. As discussed in section 4.6.3, the core of C-HTS is the process of measuring the semantic relatedness between text clusters using the explicit semantic representation of text. Additionally, as discussed in section 5.2.6, the same technique is used by the Slice Selector module to semantically represent the query of the adaptive system's request. This process of text representation is essentially based on the underlying concept space that has been built from Wikipedia. Since Wikipedia is available in many languages[18], building concept spaces for other languages can be done as described in section 4.2. Thus, moving C-HTS from one language to another can be done easily. The only step that needs to be changed is the morphological analysis (section 4.4.1) to filter out and stem the prominent terms in text. This step is relatively easy to implement as there has been a large volume of work completed on morphological analysis for languages other

[18] As of April 2018, there are 298 Wikipedias of which 288 are active and 10 are not: https://en.wikipedia.org/wiki/List_of_Wikipedias [Accessed: April 08, 2018]

than English (Manning et al., 2014).

Moreover, because CROCC is designed using a flexible architecture, it allows for plugging-in, removing, enabling, or disabling alternative components or algorithms used by the service. Thus, other components of the service (Content Harvester, Content Pruner, etc.) can be replaced, if required, based on the language of the target content resources. Hence, the initial extension of this research would consist of extending CROCC to provide multilingual content-supply services. This in turn would allow CROCC to incorporate and supply a wider range of content resources.

7.3.2 Domain-Specific Concept Space

This research proposed a novel approach for structuring content resources based on their semantic representation. This semantic structure is further used by a novel content-supply service named CROCC. As discussed in section 4.3, the concept space used in this research is built from Wikipedia. Since Wikipedia contains millions of documents in different domains[19], this means that the concept space built from it is generic and does not focus on a specific domain. Thus, a potential piece of future work could involve the creation of a concept space using a domain-specific knowledge repository. This would involve research into the characteristics of the knowledge repository that the concept space would be built from. The task of building a concept space from a knowledge repository other than Wikipedia has already been studied (Gottron et al., 2011), and it has been shown experimentally that instead of the conceptually orthogonal Wikipedia articles, using documents from the Reuters corpus lead to a comparable performance (Anderka and Stein, 2009). To build further on this line of future work, evaluating the impact of such a concept space on the segmentation task (C-HTS) and the content-supply (CROCC) would be undertaken.

7.3.3 Integrate Different Content Annotation Tools

The implementation of the CROCC service (section 5.4) and the experiment described in Chapter 6 have primarily focused on two main elements of the request that might be sent by an adaptive system: the query and the level of granularity (number of sentences). However, as discussed in section 5.2.3, the flexibility of building the structure of a content resource using C-HTS, allows for including a variety of metadata information that gives

[19] https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia

more description to each slice produced in the structure. Additionally, the flexibility of the CROCC structure allows for plugging-in different components or algorithms as required. Thus, a potential piece of future work may involve integrating different annotation tools that can annotate slices with additional metadata information. Examples of such metadata in the education domain are, reading difficulty and pedagogical annotations. This line of future work would also include the evaluation of the newly added information in a full content adaptation scenario that would involve using an actual adaptive system.

7.4 Final Remarks

It is hoped by the author of this thesis that the proposed hierarchical text segmentation approach will be of benefit to both the research community and commercial applications. The proposed C-HTS algorithm will be useful for researchers who wish to build a structure from textual documents based on the semantic representation of text. This structure can be utilised, for example, in a semantic retrieval task where the indexing process, and the retrieval process, will be based on the semantic representation of the structure of the document not on the traditional keyword-based indexing mechanism. C-HTS would also be of benefit for content mining companies that focus on techniques for understanding the meaning of the text.

It is also hoped that the proposed content-supply service CROCC, will be of benefit to researchers who wish to build adaptive systems and/or content-supply services. Researchers could continue to contribute to new/advanced features of the service in the future, with a target of the exploitation of various other content processing and adaptation approaches. The experiment in this thesis has shown the benefits that CROCC can bring to content discoverability and reusability for adaptive systems. As a result, adaptive systems designers can employ the service to facilitate the delivery of content to their users.

References

Agerri, R., & Rigau, G. (2016). Robust multilingual Named Entity Recognition with shallow semi-supervised features. Artificial Intelligence, 238, 63–82.

Aghoutane, B., El Fazazy, K., El Bannay, O., & El Makhfi, N. (2017). Integration Strategy for the Realization of an Adaptive Hypermedia System of Natural Dyes. International Journal of Systems Engineering, 1(1), 10-16.

Agirre, E., de Lacalle, O., & Soroa, A. (2014). Random walks for knowledge-based word sense disambiguation. Computational Linguistics, 40(1), 57–84.

Agirre, E., & Stevenson, M. (2006). Knowledge Sources for WSD. In E. Agirre & P. Edmonds (Eds.), Word Sense Disambiguation: Algorithms and Applications (pp. 217–251).

Ahmadi, H., & Kong, J. (2008). Efficient Web Browsing on Small Screens. In Proceedings of the Working Conference on Advanced Visual Interfaces (pp. 23–30). New York, NY, USA: ACM.

Ahn, J., Brusilovsky, P., Grady, J., He, D., & Syn, S. Y. (2007). Open User Profiles for Adaptive News Systems: Help or Harm? In Proceedings of the 16th International Conference on World Wide Web (pp. 11–20). New York, NY, USA: ACM.

Aitken, J. S. (2002). Learning information extraction rules: An inductive logic programming approach. In ECAI (pp. 355–359).

Alani, H., Kim, S., Millard, D. E., Weal, M. J., Lewis, P. H., Hall, W., & Shadbolt, N. R. (2003). Automatic extraction of knowledge from web documents. In Workshop on Human Language Technology for the Semantic Web and Web Services, 2nd international Semantic Web Conference. Sanibel Island, Florida, USA.

Alassi, D., & Alhajj, R. (2013). Effectiveness of template detection on noise reduction and websites summarization. Information Sciences, 219, 41–72.

Alfonseca, E., & Rodríguez, P. (2003). Generating Extracts with Genetic Algorithms. In F. Sebastiani (Ed.), Advances in Information Retrieval (pp. 511–519).

Alfonseca, E., Rodríguez, P., & Pérez, D. (2007). An approach for automatic generation of adaptive hypermedia in education with multilingual knowledge discovery techniques. Computers & Education, 49(2), 495–513.

Allan, J., Carbonell, J., Doddington, G., Yamron, J., & Yang, Y. (1998). Topic Detection and Tracking Pilot Study: Final Report. In Proceedings of the DARPA Broadcast News Transcription and Understanding Worksho (pp. 194–218).

Allison, L., & Dix, T. I. (1986). A bit-string longest-common-subsequence algorithm. Information Processing Letters, 23(5), 305–310.

Anderka, M., & Stein, B. (2009). The ESA Retrieval Model Revisited. In Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 670–671). New York, NY, USA: ACM.

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Wittrock, M. C. (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, abridged edition. White Plains, NY: Longman.

Angheluta, R., De Busser, R., & Moens, M.-F. (2002). The use of topic segmentation for automatic summarization. Proceedings of the ACL-2002 Workshop on Automatic Summarization (pp. 11-12).

Arora, S., Ge, R., Halpern, Y., Mimno, D., Moitra, A., Sontag, D., Zhu, M. (2013). A practical algorithm for topic modeling with provable guarantees. In International Conference on Machine Learning (pp. 280–288).

Aroyo, L., Bra, P. De, Houben, G.-J., & Vdovjak, R. (2004). Embedding information retrieval in adaptive hypermedia: IR meets AHA! New Review of Hypermedia and Multimedia, 10(1), 53–76.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). DBpedia: A Nucleus for a Web of Open Data. The Semantic Web (pp. 722–735).

Azzopardi, L., Moshfeghi, Y., Halvey, M., Alkhawaldeh, R. S., Balog, K., Di Buccio, E., Palchowdhury, S. (2017). Lucene4IR: Developing Information Retrieval Evaluation Resources Using Lucene. SIGIR Forum, 50(2), 58–75.

Badjatiya, P., Kurisinkel, L. J., Gupta, M., & Varma, V. (2018). Attention-Based Neural Text Segmentation. Advances in Information Retrieval. Springer International Publishing, 180–193.

Bailey, C., Zalfan, M. T., Davis, H. C., Fill, K., & Conole, G. (2006). Panning for gold: designing pedagogically-inspired learning nuggets. Journal of Educational Technology & Society, 9(1), 113-122.

Bär, D., Zesch, T., & Gurevych, I. (2013). Dkpro similarity: An open source framework for text similarity. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 121–126).

Basile, P., Caputo, A., & Semeraro, G. (2014). An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers (pp. 1591–1600).

Baumgartner, R., Flesca, S., & Gottlob, G. (2001). Declarative Information Extraction, Web Crawling, and Recursive Wrapping with Lixto. Logic Programming and Nonmotonic Reasoning. Springer Berlin Heidelberg, 21–41.

Bayomi, M. (2015). A Framework to Provide Customized Reuse of Open Corpus Content for Adaptive Systems. In Proceedings of the 26th ACM Conference on Hypertext & Social Media (pp. 315–318). New York, NY, USA: ACM.

Bayomi, M., & Lawless, S. (2016). ADAPT_TCD: An Ontology-Based Context Aware Approach for Contextual Suggestion. In E. M. Voorhees (Ed.), NIST Special Publication: The Twenty-Fifth Text REtrieval Conference Proceedings (TREC 2016), Contextual Suggestion Track. Gaithersburg, Maryland, USA: National Institute for Standards and Technology, NIST Special Publication 500-321.

Bayomi, M., & Lawless, S. (2018). C-HTS: A Concept-based Hierarchical Text Segmentation approach. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). Miyazaki, Japan: European Language Resources Association (ELRA).

Bayomi, M., Levacher, K., Ghorab, M. R., Lavin, P., O'Connor, A., & Lawless, S. (2016). Towards Evaluating the Impact of Anaphora Resolution on Text Summarisation from a Human Perspective. Natural Language Processing and Information Systems: 21st

International Conference on Applications of Natural Language to Information Systems, NLDB 2016, Salford, UK. Springer International Publishing, 187–199.

Bayomi, M., Levacher, K., Ghorab, M. R., & Lawless, S. (2015). OntoSeg: A Novel Approach to Text Segmentation Using Ontological Similarity. In 2015 IEEE International Conference on Data Mining Workshop (ICDMW) (pp. 1274–1283).

Beck, C., Streicher, A., & Zielinski, A. (2014). Using Text Segmentation Algorithms for the Automatic Generation of E-Learning Courses. Lexical and Computational Semantics (* SEM 2014), 132-140.

Beeferman, D., Berger, A., & Lafferty, J. (1999). Statistical Models for Text Segmentation. Machine Learning, 34(1–3), 177–210.

Benyon, D., & Murray, D. (1993). Applying user modeling to human-computer interaction design. Artificial Intelligence Review, 7(3), 199–225.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. Scientific American, 284(5), 34–43.

Białecki, A., Muir, R., & Ingersoll, G. (2012). Apache lucene 4. In SIGIR 2012 workshop on open source information retrieval (p. 17).

Bing, L., Guo, R., Lam, W., Niu, Z.-Y., & Wang, H. (2014). Web Page Segmentation with Structured Prediction and Its Application in Web Page Classification. In Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (pp. 767–776). New York, NY, USA: ACM.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of Machine Learning Research, 3(Jan), 993–1022.

Boguraev, B. K., & Neff, M. S. (2000). Discourse segmentation in aid of document summarization. System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference, IEEE.

Bohl, O., Scheuhase, J., Sengler, R., & Winand, U. (2002). The sharable content object reference model (SCORM) - a critical review. In International Conference on Computers in Education, 950–951.

Bokaei, M. H., Sameti, H., & Liu, Y. (2016). Extractive summarization of multi-party meetings through discourse segmentation. Natural Language Engineering, 22(1), 41–72.

Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction, 6(2), 87–129.

Brusilovsky, P. (1998). Methods and Techniques of Adaptive Hypermedia. Adaptive Hypertext and Hypermedia. Springer Netherlands, 1–43.

Brusilovsky, P. (2001). Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 11(1–2), 87–110.

Brusilovsky, P. (2004). KnowledgeTree: A Distributed Architecture for Adaptive e-Learning. In Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters (pp. 104–113). New York, NY, USA: ACM.

Brusilovsky, P., Chavan, G., & Farzan, R. (2004). Social Adaptive Navigation Support for Open Corpus Electronic Textbooks. Adaptive Hypermedia and Adaptive Web-Based Systems SE - 6 (Vol. 3137). Springer Berlin Heidelberg, 24–33.

Brusilovsky, P., Eklund, J., & Schwarz, E. (1998). Web-based education for all: a tool for development adaptive courseware. Computer Networks and ISDN Systems, 30(1), 291–300.

Brusilovsky, P., & Henze, N. (2007). Open Corpus Adaptive Educational Hypermedia. The Adaptive Web SE - 22 (Vol. 4321). Springer Berlin Heidelberg, 671–696.

Brusilovsky, P., & Millán, E. (2007). User Models for Adaptive Hypermedia and Adaptive Educational Systems. The Adaptive Web: Methods and Strategies of Web Personalization. Springer Berlin Heidelberg, 3–53.

Brusilovsky, P., & Pesin, L. (1998). Adaptive navigation support in educational hypermedia: An evaluation of the ISIS-Tutor. Journal of Computing and Information Technology, 6(1), 27–38.

Brusilovsky, P., Schwarz, E., & Weber, G. (1996). ELM-ART: An intelligent tutoring system on world wide web. Intelligent Tutoring Systems: Third International Conference, ITS '96 Montréal, Canada. Springer Berlin Heidelberg, 261–269.

Buchanan, B. G., & Feigenbaum, E. (1982). Forward. In Davis, R., and Lenat, D. B., eds., Knowledge-Based Systems in Artificial Intelligence. McGraw-Hill.

Budanitsky, A., & Hirst, G. (2006). Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Comput. Linguist., 32(1), 13–47.

Bunt, A., Carenini, G., & Conati, C. (2007). Adaptive Content Presentation for the Web. The Adaptive Web: Methods and Strategies of Web Personalization. Springer Berlin Heidelberg, 409–432.

Cambria, E., Fu, J., Bisio, F., & Poria, S. (2015). AffectiveSpace 2: Enabling affective intuition for concept-level sentiment analysis. In Twenty-Ninth AAAI Conference on Artificial Intelligence (pp. 508–514).

Cambria, E., & White, B. (2014). Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]. IEEE Computational Intelligence Magazine, 9(2), 48–57.

Cañas, A. J., Valerio, A., Lalinde-Pulido, J., Carvalho, M., & Arguedas, M. (2003). Using WordNet for Word Sense Disambiguation to Support Concept Map Construction. String Processing and Information Retrieval. Springer Berlin Heidelberg, 350–359.

Carmona, C., Bueno, D., Guzman, E., & Conejo, R. (2002). SIGUE: Making Web Courses Adaptive. In P. De Bra, P. Brusilovsky, & R. Conejo (Eds.), Adaptive Hypermedia and Adaptive Web-Based Systems: Second International Conference, AH 2002 Málaga, Spain. Springer Berlin Heidelberg, 376–379.

Carroll, L. (2010). Evaluating Hierarchical Discourse Segmentation. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL (pp. 993–1001).

Carvalho, D., Çalli, Ç., Freitas, A., & Curry, E. (2014). EasyESA: A Low-effort Infrastructure for Explicit Semantic Analysis. In Proceedings of the 2014 International Conference on Posters & Demonstrations Track - Volume 1272 (pp. 177–180).

Chakrabarty, A., Pandit, O. A., & Garain, U. (2017). Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Vol. 1, pp. 1481–1491).

Chang, A. X., & Manning, C. D. (2014). TokensRegex: Defining cascaded regular expressions over tokens. Tech. Rep. CSTR (2014).

Chang, C.-H., Kayed, M., Girgis, M. R., & Shaalan, K. F. (2006). A Survey of Web Information Extraction Systems. IEEE Transactions on Knowledge and Data Engineering, 18(10), 1411–1428.

Chang, M.-W., Ratinov, L.-A., Roth, D., & Srikumar, V. (2008). Importance of Semantic Representation Dataless Classification. In AAAI (Vol. 2, pp. 830–835).

Chaves-González, J. M., & Martínez-Gil, J. (2013). Evolutionary algorithm based on different semantic similarity functions for synonym recognition in the biomedical domain. Knowledge-Based Systems, 37, 62–69.

Chesnais, P. R., Mucklo, M. J., & Sheena, J. A. (1995). The Fishwrap personalized news system. In Community Networking, 1995. Proceedings of the Second International Workshop on Integrated Multimedia Services to the Home, 275–282.

Chien, J. T., & Chueh, C. H. (2012). Topic-Based Hierarchical Segmentation. IEEE Transactions on Audio, Speech, and Language Processing, 20(1), 55–66.

Choi, F. Y. Y. (2000). Advances in Domain Independent Linear Text Segmentation. In Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference (pp. 26–33). Association for Computational Linguistics.

Choi, F. Y. Y., Wiemer-Hastings, P., & Moore, J. (2001). Latent Semantic Analysis for Text Segmentation. In Proceedings of EMNLP (pp. 109–117).

Chowdhury, G. G. (2003). Natural language processing. Annual Review of Information Science and Technology, 37(1), 51–89.

Church, K. W. (1993). Char_Align: A Program for Aligning Parallel Texts at the Character Level. In Proceedings of the 31st Annual Meeting on Association for Computational Linguistics (pp. 1–8).

Cohen, D., & Croft, W. B. (2018). A Hybrid Embedding Approach to Noisy Answer Passage Retrieval. Advances in Information Retrieval. Springer International Publishing, 127–140.

Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing (pp. 1–8).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of Machine Learning Research, 2493–2537.

Conlan, O., Staikopoulos, A., Hampson, C., Lawless, S., & O'keeffe, I. (2013). The narrative approach to personalisation. New Review of Hypermedia and Multimedia, 19(2), 132–157.

Conlan, O., & Wade, V. (2004). Evaluation of APeLS - An Adaptive eLearning Service based on the Multi-model, Metadata-driven Approach. International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. Springer, Berlin, Heidelberg, 291 – 295.

Conlan, O., Wade, V., Bruen, C., & Gargan, M. (2002). Multi-model, metadata driven approach to adaptive hypermedia services for personalized elearning. International

Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. Springer, Berlin, Heidelberg, 100–111.

Constantin, A., Pettifer, S., & Voronkov, A. (2013). PDFX: Fully-automated PDF-to-XML Conversion of Scientific Literature. In Proceedings of the 2013 ACM Symposium on Document Engineering (pp. 177–180). New York, NY, USA: ACM.

Corra, E. A., Lopes, A. A., & Amancio, D. R. (2018). Word Sense Disambiguation. Information Sciences: an International Journal 442.C (2018): 103-113.

Cowie, J., & Lehnert, W. (1996). Information Extraction. Communications of the ACM, 39(1), 80–91.

Cruz, F., Troyano, J., & Enríquez, F. (2006). Supervised TextRank. Advances in Natural Language Processing SE . Springer Berlin Heidelberg, 632–639.

Curran, J. R. (2002). Ensemble Methods for Automatic Thesaurus Extraction. In Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing , 222–229).

da Silva, D. P., Van Durm, R., Duval, E., & Olivié, H. (1998). Concepts and documents for adaptive educational hypermedia: a model and a prototype. In Second workshop on Adaptive Hypertext and Hypermedia, Ninth ACM Conference on Hypertext and Hypermedia, Pittsburgh, USA (pp. 35–43).

Dagger, D., Wade, V., & Conlan, O. (2002). Towards a Standards-based Approach to e-Learning Personalization using Reusable Learning Objects. World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2002. AACE, 210–217.

Daiber, J., Jakob, M., Hokamp, C., & Mendes, P. N. (2013). Improving Efficiency and Accuracy in Multilingual Entity Extraction. In Proceedings of the 9th ACM International Conference on Semantic Systems (I-Semantics).

De Bra, P., Aerts, A., Berden, B., de Lange, B., Rousseau, B., Santic, T., Stash, N. (2003). AHA! The Adaptive Hypermedia Architecture. In Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia (pp. 81–84). New York, NY, USA: ACM.

De Bra, P., & Calvi, L. (1997). Creating Adaptive Hyperdocuments for and on the Web. In WebNet 1997.

De Bra, P., & Calvi, L. (1998). AHA: a generic adaptive hypermedia system. In Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia (pp. 5–12).

De Bra, P., Houben, G.-J., & Wu, H. (1999). AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. In Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia (pp. 147–156). New York, NY, USA: ACM.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6), 391– 407.

Dieberger, A., & Guzdial, M. (2003). CoWeb — Experiences with Collaborative Web Spaces. From Usenet to CoWebs. Springer London, 155–166.

Dimitrova, V. (2003). STyLE-OLM: Interactive open learner modelling. International Journal of Artificial Intelligence in Education, 13(1), 35–78.

Dimitrova, V., & Brna, P. (2016). From Interactive Open Learner Modelling to Intelligent Mentoring: STyLE-OLM and Beyond. International Journal of Artificial Intelligence in Education, 26(1), 332–349.

Dolog, P., Henze, N., Nejdl, W., & Sintek, M. (2004). The Personal Reader: Personalizing and Enriching Learning Resources Using Semantic Web Technologies. Adaptive Hypermedia and Adaptive Web-Based Systems. Springer Berlin Heidelberg, 85–94.

Du, L., Buntine, W. L., & Johnson, M. (2013). Topic Segmentation with a Structured Topic Model. In Proceedings of the North American Chapter of the Association for Computational Linguistics Conference. 2013 (pp. 190 –200).

Egozi, O., Markovitch, S., & Gabrilovich, E. (2011). Concept-Based Information Retrieval Using Explicit Semantic Analysis. ACM Transactions of Information Systems, 29(2), 8.

Eisenstein, J. (2009). Hierarchical Text Segmentation from Multi-Scale Lexical Cohesion. Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 353–361.

Eisenstein, J., & Barzilay, R. (2008). Bayesian Unsupervised Topic Segmentation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 334–343). Stroudsburg, PA, USA: Association for Computational Linguistics.

Elavarasi, S. A., Akilandeswari, J., & Menaga, K. (2014). A survey on semantic similarity measure. International Journal of Research in Advent Technology, 2(3), 389–398.

Fader, A., Soderland, S., & Etzioni, O. (2011). Identifying Relations for Open Information Extraction. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 1535–1545). Stroudsburg, PA, USA: Association for Computational Linguistics.

Fang, Y., Xie, X., Zhang, X., Cheng, R., & Zhang, Z. (2018). STEM: a suffix tree-based method for web data records extraction. Knowledge and Information Systems, 55(2), 305–331.

Farrah, S., Manssouri, H. El, Ziyati, E. H., & Ouzzif, M. (2018). An hybrid approach to improve part of speech tagging system. In 2018 International Conference on Intelligent Systems and Computer Vision (ISCV) (pp. 1–6).

Farrell, R. G., Liburd, S. D., & Thomas, J. C. (2004). Dynamic Assembly of Learning Objects. In Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters (pp. 162–169). New York, NY, USA: ACM.

Feigenbaum, L., Herman, I., Hongsermeier, T., Neumann, E., & Stephens, S. (2007). The semantic web in action. Scientific American, 297(6), 90–97.

Feng, V. W., & Hirst, G. (2012). Text-level Discourse Parsing with Rich Linguistic Features. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: (pp. 60–68).

Fernández, M., Cantador, I., López, V., Vallet, D., Castells, P., & Motta, E. (2011). Semantically enhanced Information Retrieval: An ontology-based approach. Web Semantics: Science, Services and Agents on the World Wide Web, 434–452.

Ferschke, O., Zesch, T., & Gurevych, I. (2011). Wikipedia Revision Toolkit: Efficiently Accessing Wikipedia's Edit History. In Proceedings of the ACL-HLT 2011 System Demonstrations (pp. 97–102).

Finin, T. W. (1989). GUMS - A General User Modeling Shell. User Models in Dialog Systems. Springer Berlin Heidelberg, 411–430.

Fiqri, M., & Nurjanah, D. (2017). Graph-based domain model for adaptive learning path recommendation. In 2017 IEEE Global Engineering Education Conference (EDUCON) (pp. 375–380).

Fiszman, M., & Rindflesch, T. C. (2003). Abstraction Summarization for Managing the Biomedical Research Literature. Proceedings of the HLT-NAACL workshop on computational lexical semantics. Association for Computational Linguistics, 76-83.

Francis, W. N. (1964). A Standard Sample of Present-Day English for Use With Digital Computers. Report to the U.S Office of Education on Cooperative Research Project No. E-007. Brown University.

Fröhlich, P., Nejdl, W., & Wolpers, M. (1998). KBS-Hyperbook-An Open Hyperbook System for Education. In Proceedings of the ED-MEDIA World Conference on Educational Multimedia and Hypermedia.

Gabbard, R., DeYoung, J., Lignos, C., Freedman, M., & Weischedel, R. (2018). Combining rule-based and statistical mechanisms for low-resource named entity recognition. Machine Translation, 32(1), 31–43.

Gabrilovich, E., & Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In IJcAI (Vol. 7, pp. 1606–1611).

Gabrilovich, E., & Markovitch, S. (2009). Wikipedia-based semantic interpretation for natural language processing. Journal of Artificial Intelligence Research, 34, 443–498.

Galley, M., McKeown, K., Fosler-Lussier, E., & Jing, H. (2003). Discourse Segmentation of Multi-party Conversation. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (pp. 562–569).

Gandara, R. A., Nurjanah, D., & Ciptasari, R. W. (2014). Integrating cognitively-oriented and pedagogically-oriented methods in adaptive educational hypermedia. In 2014 International Conference on Data and Software Engineering (ICODSE) (pp. 1–6).

Gao, J. B., Zhang, B. W., & Chen, X. H. (2015). A simplified edge-counting method for measuring semantic similarity of concepts. In 2015 International Conference on Machine Learning and Cybernetics (ICMLC) (pp. 176–181).

Ghauth, K. I., & Abdullah, N. A. (2011). The Effect of Incorporating Good Learners' Ratings in e-Learning Content-based Recommender System. Journal of Educational Technology & Society, 14(2), 248–257.

Ghorab, M. R. (2014). A Framework for the Delivery and Evaluation of Personalised Multilingual Information Retrieval. PhD thesis, Trinity College Dublin.

Ghorab, M. R., Zhou, D., O'Connor, A., & Wade, V. (2013). Personalised Information Retrieval: survey and classification. User Modeling and User-Adapted Interaction, 23(4), 381–443.

Givoni, I. E., Chung, C., & Frey, B. J. (2011). Hierarchical Affinity Propagation. In Uncertainty in AI, Proceedings of the Twenty-Seventh Conference (pp. 238–246).

Glavaš, G., Nanni, F., & Ponzetto, S. P. (2016). Unsupervised text segmentation using semantic relatedness graphs. In *SEM 2016: The Fifth Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics, 125–130.

Goldwater, S., Griffiths, T. L., & Johnson, M. (2006). Contextual Dependencies in Unsupervised Word Segmentation. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (pp. 673–680).

Gonzalez-Barahona, J. M., Dimitrova, V., Chaparro, D., Tebb, C., Romera, T., Canas, L., Kleanthous, S. (2006). Towards Community-Driven Development of Educational Materials: The Edukalibre Approach. Innovative Approaches for Learning and Knowledge Sharing. Springer Berlin Heidelberg, 125–139.

Gottron, T., Anderka, M., & Stein, B. (2011). Insights into Explicit Semantic Analysis. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management (pp. 1961–1964). New York, NY, USA: ACM.

Grosz, B. J., & Sidner, C. L. (1986). Attention, Intentions, and the Structure of Discourse. Computational linguistics., 12(3), 175–204.

Gurevych, I., Müller, C., & Zesch, T. (2007). What to be?-electronic career guidance based on semantic relatedness. In Annual Meeting-Association for Computational Linguistics (Vol. 45, pp. 1032–1039).

Habash, N., & Rambow, O. (2005). Arabic Tokenization, Part-of-speech Tagging and Morphological Disambiguation in One Fell Swoop. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (pp. 573–580).

Habibi, M., Weber, L., Neves, M., Wiegandt, D. L., & Leser, U. (2017). Deep learning with word embeddings improves biomedical named entity recognition. Bioinformatics, 33(14), i37–i48.

Hajeer, S. I., Ismail, R. M., Badr, N. L., & Tolba, M. F. (2017). A New Stemming Algorithm for Efficient Information Retrieval Systems and Web Search Engines. Multimedia Forensics and Security: Foundations, Innovations, and Applications. Springer International Publishing, 117–135.

Halasz, F., & Schwartz, M. (1990). The Dexter Reference Model. In NIST Hypertext Standardization Workshop (pp. 95–133).

Halasz, F., & Schwartz, M. (1994). The Dexter Hypertext Reference Model. Communications of the ACM, 37(2), 30–39.

Hamp, B., & Feldweg, H. (1997). GermaNet- a Lexical-Semantic Net for German. Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications.

Han, E.-H. (Sam), & Karypis, G. (2000). Centroid-Based Document Classification: Analysis and Experimental Results. Principles of Data Mining and Knowledge Discovery. Springer Berlin Heidelberg, 424–431.

Hargood, C., Millard, D., & Weal, M. (2011). Measuring Narrative Cohesion: A Five Variables Approach. Narrative and Hypertext Workshop at Hypertext 11.

Hassen, F., & Amel, G. T. (2017). An efficient synchronous indexing technique for full-text retrieval in distributed databases. Procedia Computer Science, 112, 811–821.

Hearst, M. A. (1994). Multi-paragraph Segmentation of Expository Text. In Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics (pp. 9–16).

Hearst, M. A. (1997). TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages. Computational linguistics, 23(1), 33–64.

Henze, N., & Nejdl, W. (2001). Adaptation in open corpus hypermedia. International Journal of Artificial Intelligence in Education, 12(4), 325–350.

Henze, N., & Nejdl, W. (2002). Knowledge modeling for open adaptive hypermedia. International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. Springer, Berlin, Heidelberg,174–183.

Hill, F., Reichart, R., & Korhonen, A. (2015). SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. Computational Linguistics, 41(4), 665–695.

Hsueh, P.-Y., D. Moore, J., & Renals, S. (2006). Automatic segmentation of multiparty dialogue. 11th Conference of the European Chapter of the Association for Computational Linguistics.

Hutchins, J. (2005). The first public demonstration of machine translation: the Georgetown-IBM system, 7th January 1954. In AMTA Conference.

Jain, A. K., & Dubes, R. C. (1988). Algorithms for clustering data. Prentice hall Englewood Cliffs.

Jamil, N., Ramli, M. I., & Seman, N. (2015). Sentence boundary detection without speech recognition: A case of an under-resourced language. Journal of Electrical Systems, 11(3).

Janati, S. El, Maach, A., & Ghanami, D. El. (2018). SMART Education Framework for Adaptation Content Presentation. Procedia Computer Science, 127, 436–443.

Jiang, Y., Bai, W., Zhang, X., & Hu, J. (2017). Wikipedia-based information content and semantic similarity computation. Information Processing & Management, 53(1), 248–265.

Jiang, Y., Zhang, X., Tang, Y., & Nie, R. (2015). Feature-based approaches to semantic similarity assessment of concepts using Wikipedia. Information Processing & Management, 51(3), 215–234.

Joel, P., Nada, L., & Dunja, M. (2004). A rule based approach to word lemmatization. In Proceedings of the 7th International Multi-Conference Information Society IS.

John, A. K., Di Caro, L., & Boella, G. (2017). Text Segmentation with Topic Modeling and Entity Coherence. Proceedings of the 16th International Conference on Hybrid Intelligent Systems (HIS 2016). Springer International Publishing, 175–185.

Jungwirth, M., & Hanbury, A. (2018). Replicating an experiment in cross-lingual information retrieval with explicit semantic analysis. In CLEF (pp. 73–1613).

Jurafsky, D. (2000). Speech and language processing: An introduction to natural language processing. Computational Linguistics, and Speech Recognition.

Kan, M.-Y., Klavans, J. L., & McKeown, K. (1998). Linear Segmentation and Segment Significance. In 6th international Workshop of Very Large Corpora (WVLC-6) (pp. 197–205).

Kardan, A. A., Aziz, M., & Shahpasand, M. (2015). Adaptive systems: a content analysis on technical side for e-learning environments. Artificial Intelligence Review, 44(3), 365–391.

Karimi, M., Jannach, D., & Jugovac, M. (2018). News recommender systems – Survey and roads ahead. Information Processing & Management, 54(6), 1203–1227.

Kazantseva, A., & Szpakowicz, S. (2014). Hierarchical Topical Segmentation with Affinity Propagation. Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics (pp. 37–47).

Keeffe, I. O., Connor, A. O., Cass, P., Lawless, S., Wade, V., Okeeffe, I., Lawless, S. (2012). Linked Open Corpus Models , Leveraging the Semantic Web for Adaptive Hypermedia. Proceedings of the 23rd ACM conference on Hypertext and social media, 321–322.

Kim, J. W., & Cho, S.-H. (2014). Effectively detecting topic boundaries in a news video by using wikipedia. International Journal of Software Engineering and Its Applications, 8(6), 229–240.

Knutov, E., Bra, P. De, & Pechenizkiy, M. (2009). AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. New Review of Hypermedia and Multimedia, 15(1), 5–38.

Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M. Lee, R. (2009). Media Meets Semantic Web -- How the BBC Uses DBpedia and Linked Data to Make Connections. The Semantic Web: Research and Applications. Springer Berlin Heidelberg, 723–737.

Koch, N., & Wirsing, M. (2002). The Munich reference model for adaptive hypermedia applications. International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. Springer Berlin Heidelberg, 213–222.

Kohlschütter, C., & Nejdl, W. (2008). A Densitometric Approach to Web Page Segmentation. In Proceedings of the 17th ACM Conference on Information and Knowledge Management (pp. 1173–1182). New York, NY, USA: ACM.

Labadié, A., & Prince, V. (2008). Finding Text Boundaries and Finding Topic Boundaries: Two Different Tasks? Advances in Natural Language Processing. Springer Berlin Heidelberg, 260–271.

Labutov, I., Huang, Y., Brusilovsky, P., & He, D. (2017). Semi-Supervised Techniques for Mining Learning Outcomes and Prerequisites. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 907–915). New York, NY, USA: ACM.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. Discourse Processes, 25(2–3), 259–284.

Lawless, S. (2009). Leveraging Content from Open Corpus Sources for Technology Enhanced Learning. PhD thesis, Trinity College Dublin.

Lawless, S., Lavin, P., Bayomi, M., Cabral, J. P., & Ghorab, M. R. (2015). Text Summarization and Speech Synthesis for the Automated Generation of Personalized Audio Presentations. Natural Language Processing and Information Systems. Springer International Publishing, 307–320.

Leacock, C., & Chodorow, M. (1998). Combining local context and WordNet similarity for word sense identification. MIT Press, Cambridge, MA, 49(2), 265–283.

Leoncini, A., Sangiacomo, F., Gastaldo, P., & Zunino, R. (2012). A Semantic-Based Framework for Summarization and Page Segmentation in Web Mining. Theory and Applications for Advanced Text Mining. InTech 2012.

Lesk, M. (1986). Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In Proceedings of the 5th Annual International Conference on Systems Documentation (pp. 24–26). New York, NY, USA: ACM.

Leung, Y., Zhang, J.-S., & Xu, Z.-B. (2000). Clustering by scale-space filtering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(12), 1396–1410.

Levacher, K. (2014). Slicepedia Open Corpus Slicing for Adaptive Web Systems. PhD thesis, Trinity College Dublin.

Levacher, K., Hynes, É., Lawless, S., O'Connor, A., & Wade, V. (2009). A framework for content preparation to support open-corpus adaptive hypermedia. In International Workshop on Dynamic and Adaptive Hypertext: generic Frameworks, approaches and Techniques (pp. 1–11).

Levacher, K., Lawless, S., & Wade, V. (2011). A Proposal for the Evaluation of Adaptive Content Retrieval, Modification and Delivery. In Proceedings of the First Workshop on Personalised Multilingual Hypertext Retrieval (pp. 18–25). New York, NY, USA: ACM.

Levacher, K., Lawless, S., & Wade, V. (2012a). Slicepedia: Automating the Production of Educational Resources from Open Corpus Content. 21st Century Learning for 21st Century Skills: 7th European Conference of Technology Enhanced Learning, EC-TEL. Springer Berlin Heidelberg, 407–412.

Levacher, K., Lawless, S., & Wade, V. (2012b). Slicepedia: Providing Customized Reuse of Open-web Resources for Adaptive Hypermedia. In Proceedings of the 23rd ACM Conference on Hypertext and Social Media (pp. 23–32). New York, NY, USA: ACM.

Levacher, K., Lawless, S., & Wade, V. (2012c). Slicepedia: towards long tail resource production through open corpus reuse. In Advances in Web-Based Learning-ICWL 2012. Springer Berlin Heidelberg, 109–119.

Levacher, K., Lawless, S., & Wade, V. (2014). Slicepedia: Content-agnostic slicing resource production for adaptive hypermedia. Computer Science and Information Systems, 11(1), 393–417.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In Soviet physics doklady (Vol. 10, pp. 707–710).

Li, P., Wang, H., Li, H., & Wu, X. (2018). Employing Semantic Context for Sparse Information Extraction Assessment. ACM Transactions on Knowledge Discovery from Data (TKDD) 12(5), 54:1--54:36.

Liang, M. E., Guerra, J., & Brusilovsky, P. (2012). Building multi-layer social knowledge maps with google maps API. In CEUR Workshop Proceedings (Vol. 872), University of Pittsburgh.

Lin, D. (1998). An information-theoretic definition of similarity. In ICML (Vol. 98, pp. 296–304).

Llopis, F., Ferrández, A., & Vicedo, J. (2002). Text Segmentation for Efficient Information Retrieval. Computational Linguistics and Intelligent Text Processing. Springer Berlin Heidelberg, 373–380.

Lu, M., Sun, X., Wang, S., Lo, D., & Duan, Y. (2015). Query expansion via WordNet for effective code search. In 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER) (pp. 545–549).

Malioutov, I., & Barzilay, R. (2006). Minimum Cut Model for Spoken Lecture Segmentation. Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. Association for Computational Linguistics, (pp. 25–32).

Malmasi, S., Dras, M., Johnson, M., Du, L., & Wolska, M. (2017). Unsupervised Text Segmentation Based on Native Language Characteristics. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Vol. 1, pp. 1457–1469).

Mann, W. C., & Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. Text-Interdisciplinary Journal for the Study of Discourse, 8(3), 243–281.

Manning, C. D. (1998). Rethinking Text Segmentation Models: An Information Extraction Case Study. Technical Report SULTRY-98-07-01, University of Sydney.

Manning, C. D., Raghavan, P., Schütze, H., & others. (2008). Introduction to information retrieval (Vol. 1). Cambridge university press.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55–60).

Masumura, R., Asami, T., Masataki, H., Sadamitsu, K., Nishida, K., & Higashinaka, R. (2017). Hyperspherical Query Likelihood Models with Word Embeddings. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Vol. 2, pp. 210–216).

Mathew, A. B. (2018). Data allocation optimization for query processing in graph databases using Lucene. Computers & Electrical Engineering.

Maycock, K. W., & Keating, J. G. (2017). The impact of an automated learning component against a traditional lecturing environment. Journal of Computer Assisted Learning, 33(6), 597–605.

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in Translation: Contextualized Word Vectors. Advances in Neural Information Processing Systems 30 (pp. 6294–6305).

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal, 5(4), 1093–1113.

Mendes, P. N., Jakob, M., García-Silva, A., & Bizer, C. (2011). DBpedia Spotlight: Shedding Light on the Web of Documents. In Proceedings of the 7th International Conference on Semantic Systems (pp. 1–8). New York, NY, USA: ACM.

Meng, R., Zhao, Z., Chi, Y., & He, D. (2017). Automatic Course Website Discovery from Search Engine Results. IConference 2017 Proceedings Vol. 2.

Meštrović, A., & Calì, A. (2017). An Ontology-Based Approach to Information Retrieval. Semantic Keyword-Based Search on Structured Data Sources. Springer International Publishing, 150–156.

Meyer, M., Rensing, C., & Steinmetz, R. (2011). Multigranularity Reuse of Learning Resources. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 7(1), 1:1--1:23.

Michaelides, D. T., Millard, D. E., Weal, M. J., & DeRoure, D. (2002). Auld Leaky: A Contextual Open Hypermedia Link Server. Hypermedia: Openness, Structural Awareness, and Adaptivity. Springer Berlin Heidelberg, 59–70.

Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing Order into Texts. Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 404–411). Association for Computational Linguistics.

Mikheev, A. (2000). Tagging Sentence Boundaries. In Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference (pp. 264–271).

Millard, D. E., Alani, H., Kim, S., Weal, M. J., Lewis, P., Hall, W., Shadbolt, N. (2003). Generating adaptive hypertext content from the semantic web. 1st International Workshop on Hypermedia and the Semantic Web, 1–9.

Millard, D. E., Moreau, L., Davis, H. C., & Reich, S. (2000). FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability Between Hypertext Domains. In Proceedings of the Eleventh ACM on Hypertext and Hypermedia (pp. 93–102). New York, NY, USA: ACM.

Miller, G. A. (1995). WordNet: A Lexical Database for English. Communications of the ACM, 38(11), 39–41.

Misra, H., Yvon, F., Jose, J. M., & Cappe, O. (2009). Text Segmentation via Topic Modeling: An Analytical Study. In Proceedings of the 18th ACM Conference on Information and Knowledge Management (pp. 1553–1556).

Mitra, B., & Craswell, N. (2017). Neural Text Embeddings for Information Retrieval. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (pp. 813–814).

Mladenic, D. (2002). Learning Word Normalization Using Word Suffix and Context from Unlabeled Data. In Proceedings of the Nineteenth International Conference on Machine Learning (pp. 427–434).

Mochihashi, D., Yamada, T., & Ueda, N. (2009). Bayesian Unsupervised Word Segmentation with Nested Pitman-Yor Language Modeling. 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, 100–108.

Mödritscher, F., Barrios, V. M. G., & Gütl, C. (2004). Enhancement of SCORM to support adaptive E-Learning within the Scope of the Research Project AdeLE. Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2004, 2499–2505.

Mohit, B. (2014). Named Entity Recognition. In I. Zitouni (Ed.), Natural Language Processing of Semitic Languages. Springer Berlin Heidelberg, 221–245.

Monge, A. E., & Elkan, C. (1996). The Field Matching Problem Algorithms and Applications. In KDD (pp. 267–270).

Morris, J., & Hirst, G. (1991). Lexical Cohesion Computed by Thesaural Relations As an Indicator of the Structure of Text. Computational linguistics, 21–48.

Moura, E. S., David, F., Berthier, R., da Silva Altigran S., & André, G. M. (2010). Using structural information to improve search in Web collections. Journal of the American Society for Information Science and Technology, 61(12), 2503–2513.

Moussallem, D., Wauer, M., & Ngomo, A.-C. N. (2018). Machine Translation using Semantic Web Technologies: A Survey. Journal of Web Semantics, 51, 1–19.

Mullen, L. A., Benoit, K., Keyes, O., Selivanov, D., & Arnold, J. (2018). Fast, Consistent Tokenization of Natural Language Text. Journal of Open Source Software, 3(23), 655.

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. Lingvisticæ Investigationes, 30(1), 3–26.

Naili, M., Habacha Chaibi, A., & Hajjami Ben Ghezala, H. (2016). Exogenous approach to improve topic segmentation. International Journal of Intelligent Computing and Cybernetics, 9(2), 165–178.

Najar, A. S., Mitrovic, A., & McLaren, B. M. (2016). Learning with intelligent tutors and worked examples: selecting learning activities adaptively leads to better learning outcomes than a fixed curriculum. User Modeling and User-Adapted Interaction, 26(5), 459–491.

Navigli, R. (2009). Word Sense Disambiguation: A Survey. ACM computing surveys (CSUR), 41(2), 10:1--10:69.

Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Risch, T. (2002). EDUTELLA: A P2P Networking Infrastructure Based on RDF. In Proceedings of the 11th International Conference on World Wide Web (pp. 604–615).

Nguyen, T. (2018). Java Spring Framework in developing the Knowledge Article Management application: A brief guide to use Spring Framework. PhD Thesis.

Onal, K. D., Zhang, Y., Altingovde, I. S., Rahman, M. M., Karagoz, P., Braylan, A., Lease, M. (2018). Neural information retrieval: at the end of the early years. Information Retrieval Journal, 21(2), 111–182.

Pak, I., & Teh, P. L. (2018). Text Segmentation Techniques: A Critical Review. Innovative Computing, Optimization and Its Applications: Modelling and Simulations Springer International Publishing, 167–181.

Pallottelli, S., Tasso, S., Pannacci, N., Costantini, A., & Lago, N. F. (2010). Distributed and Collaborative Learning Objects Repositories on Grid Networks., Computational Science and Its Applications -- ICCSA 2010. Springer Berlin Heidelberg, 29–40.

Penadés, M. C., Martí, P., Canós, J. H., & Gómez, A. (2014). Product Line-based customization of e-Government documents. PEGOV 2014: Personalization in e-Government Services, Data and Applications (Vol. 1181).

Pevzner, L., & Hearst, M. A. (2002). A Critique and Improvement of an Evaluation Metric for Text Segmentation. Computational Linguistics, 28(1), 19–36.

Philpot, A., Hovy, E., & Pantel, P. (2005). The omega ontology. Proceedings of OntoLex 2005-Ontologies and Lexical Resources.

Porter, M. F. (1980). An algorithm for suffix stripping. Program, 14(3), 130–137.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Vesely, K. (2011). The Kaldi Speech Recognition Toolkit.

Prince, V., & Labadié, A. (2007). Text Segmentation Based on Document Understanding for Information Retrieval. Natural Language Processing and Information Systems. Springer Berlin Heidelberg, 295–304.

Priya, M., & Kumar, C. A. (2015). A survey of state of the art of Ontology construction and merging using formal concept analysis. Indian Journal of Science and Technology, 8(24).

Prokofyev, R., Demartini, G., Boyarsky, A., Ruchayskiy, O., & Cudré-Mauroux, P. (2013). Ontology-Based Word Sense Disambiguation for Scientific Literature. Advances in Information Retrieval. Springer Berlin Heidelberg, 594–605.

Rada, R., Mili, H., Bicknell, E., & Blettner, M. (1989). Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man, and Cybernetics, 19(1), 17–30.

Rafferty, A. N., & Manning, C. D. (2008). Parsing three German treebanks: Lexicalized and unlexicalized baselines. In Proceedings of the Workshop on Parsing German. Association for Computational Linguistics, 2008. (pp. 40–46).

Raganato, A., Bovi, C. D., & Navigli, R. (2017). Neural sequence learning models for word sense disambiguation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 1156–1167).

Reynar, J. C. (1994). An Automatic Method of Finding Topic Boundaries. In Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics (pp. 331–333).

Reynar, J. C., & Ratnaparkhi, A. (1997). A Maximum Entropy Approach to Identifying Sentence Boundaries. In Proceedings of the Fifth Conference on Applied Natural Language Processing (pp. 16–19).

Riahi, F., Zolaktaf, Z., Shafiei, M., & Milios, E. (2012). Finding Expert Users in Community Question Answering. In Proceedings of the 21st International Conference on World Wide Web (pp. 791–798).

Riedl, M. (2016). Unsupervised Methods for Learning and Using Semantics of Natural Language, PhD thesis. Technische Universität.

Riedl, M., & Biemann, C. (2012a). Text segmentation with topic models. Journal for Language Technology and Computational Linguistics, 27(1), 47–69.

Riedl, M., & Biemann, C. (2012b). TopicTiling: A Text Segmentation Algorithm Based on LDA. In Proceedings of ACL 2012 Student Research Workshop (pp. 37–42).

Şah, M., & Hall, W. (2013). A personalized semantic portal for enhanced user support. New Review of Hypermedia and Multimedia, 19(1), 25–60.

Şah, M., & Wade, V. (2010). Automatic Metadata Extraction from Multilingual Enterprise Content. In Proceedings of the 19th ACM International Conference on Information and Knowledge Management (pp. 1665–1668).

Şah, M., & Wade, V. (2012). Automatic metadata mining from multilingual enterprise content. Web Semantics: Science, Services and Agents on the World Wide Web, 11, 41–62.

Sakahara, M., Okada, S., & Nitta, K. (2014). Domain-Independent Unsupervised Text Segmentation for Data Management. In 2014 IEEE International Conference on Data Mining Workshop (pp. 481–487).

Salton, G. (1989). Automatic text processing: The transformation, analysis, and retrieval of Information by Computer. Addison-Wesley.

Salton, G., & McGill, M. J. (1986). Introduction to Modern Information Retrieval. New York, NY, USA: McGraw-Hill, Inc.

Sánchez, D., Isern, D., & Millan, M. (2011). Content annotation for the semantic web: an automatic web-based approach. Knowledge and Information Systems, 27(3), 393–418.

Santos, C. D., & Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In Proceedings of the 31st International Conference on Machine Learning (ICML-14) (pp. 1818–1826).

Sathiyamurthy, K., & Geetha, T. V. (2011). Topic Segmentation and Evaluation Measures for E-learning based on Domain and Pedagogical Ontology. International Journal of Computer Applications, 26(6), 5–10.

Savić, G., Segedinac, M., Milenković, D., Hrin, T., & Segedinac, M. (2018). A model-driven approach to e-course management. Australasian Journal of Educational Technology, 34(1).

Selvalakshmi, B., & Subramaniam, M. (2018). Intelligent ontology based semantic information retrieval using feature selection and classification. Cluster Computing.

Shah, R., & Jain, S. (2014). Ontology-based information extraction: An overview and a study of different approaches. International Journal of Computer Applications, 87(4).

Shen, L., Satta, G., & Joshi, A. (2007). Guided learning for bidirectional sequence classification. In Proceedings of the 45th annual meeting of the association of computational linguistics (pp. 760–767).

Shima, H. (2014). WS4J WordNet Similarity for Java. Retrieved from https://code.google.com/archive/p/ws4j/ [Accessed: January 22, 2018]

Singhal, A. (2001). Modern information retrieval: A brief overview. IEEE Data Eng. Bull., 24(4), 35–43.

Slaney, M., & Ponceleon, D. (2001). Hierarchical segmentation finding changes in a text signal. In Proc. of the SIAM Text Mining 2001 Workshop, 6–13.

Sleiman, H. A., & Corchuelo, R. (2013). A Survey on Region Extractors from Web Documents. IEEE Transactions on Knowledge and Data Engineering, 25(9), 1960–1981.

Slimani, T., Yaghlane, B. Ben, & Mellouli, K. (2006). A new similarity measure based on edge counting. Proceedings of the World Academy of Science, Engineering and Technology, 17, 3.

Smith, A., & Blandford, A. (2003). MLTutor: An application of machine learning algorithms for an adaptive web-based information system. International Journal of Artificial Intelligence in Education, 13(2–4), 235–261.

Smits, D., & De Bra, P. (2011). GALE: A Highly Extensible Adaptive Hypermedia Engine. In Proceedings of the 22Nd ACM Conference on Hypertext and Hypermedia (pp. 63–72).

Song, F., Darling, W., Duric, A., & Kroon, F. (2011). An Iterative Approach to Text Segmentation. In P. Clough, C. Foley, C. Gurrin, G. F. Jones, W. Kraaij, H. Lee, & V. Mudoch (Eds.), Advances in Information Retrieval. Springer Berlin Heidelberg, . 629–640.

Sosnovsky, S., & Brusilovsky, P. (2005). Layered evaluation of topic-based adaptation to student knowledge. In Proceedings of Fourth Workshop on the Evaluation of Adaptive Systems at 10th International User Modeling Conference, UM (pp. 47–56).

Sosnovsky, S., Hsiao, I.-H., & Brusilovsky, P. (2012). Adaptation ``in the Wild'': Ontology-Based Personalization of Open-Corpus Learning Material. 21st Century Learning for 21st Century Skills: 7th European Conference of Technology Enhanced Learning, EC-TEL 2012. Springer Berlin Heidelberg, 425–431.

Staikopoulos, A., O'Keeffe, I., Rafter, R., Walsh, E., Yousuf, B., Conlan, O., & Wade, V. (2012). AMASE: A framework for composing adaptive and Personalised Learning Activities on the Web. In Advances in Web-Based Learning-ICWL 2012 (pp. 190–199).

Staikopoulos, A., O'Keeffe, I., Rafter, R., Walsh, E., Yousuf, B., Conlan, O., & Wade, V. (2014). AMASE: A framework for supporting personalised activity-based learning on the web. Computer Science and Information Systems, 343–367.

Stanovsky, G., Michael, J., Zettlemoyer, L., & Dagan, I. (2018). Supervised Open Information Extraction. In Proceedings of The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT), 885–895.

Steichen, B. (2012). Adaptive Retrieval, Composition & Presentation of Closed-Corpus and Open-Corpus Information. PhD thesis.Trinity College Dublin.

Steichen, B., Ashman, H., & Wade, V. (2012). A comparative survey of Personalised Information Retrieval and Adaptive Hypermedia techniques. Information Processing & Management, 48(4), 698–724.

Steichen, B., Lawless, S., O'Connor, A., & Wade, V. (2009). Dynamic Hypertext Generation for Reusing Open Corpus Content. In Proceedings of the 20th ACM Conference on Hypertext and Hypermedia (pp. 119–128).

Steichen, B., O'Connor, A., & Wade, V. (2011). Personalisation in the Wild: Providing Personalisation Across Semantic, Social and Open-web Resources. In Proceedings of the 22Nd ACM Conference on Hypertext and Hypermedia (pp.73–82).

Steichen, B., & Wade, V. (2010). Adaptive retrieval and composition of socio-semantic content for personalised customer care. In International Workshop on Adaptation in Social and Semantic Web (pp. 1–10).

Stokes, N., Carthy, J., & Smeaton, A. F. (2004). SeLeCT: a lexical cohesion based news story segmentation system. AI Communications, 17(1), 3–12.

Stratos, K., Collins, M., & Hsu, D. (2016). Unsupervised part-of-speech tagging with anchor hidden markov models. Transactions of the Association for Computational Linguistics, 4, 245–257.

Tasso, S., Pallottelli, S., Gervasi, O., Rui, M., & Laganà, A. (2018). Sharing Learning Objects Between Learning Platforms and Repositories. Computational Science and Its Applications. Springer International Publishing, 804–816.

Tasso, S., Pallottelli, S., Rui, M., & Laganá, A. (2014). Learning Objects Efficient Handling in a Federation of Science Distributed Repositories. Computational Science and Its Applications -- ICCSA. Springer International Publishing, 615–626.

Thaker, K., Huang, Y., Brusilovsky, P., & Daqing, H. (2018). Dynamic Knowledge Modeling with Heterogeneous Activities for Adaptive Textbooks. In The 11th International Conference on Educational Data Mining (pp. 592–595).

Tsunoo, E., Bell, P., & Renals, S. (2017). Hierarchical recurrent neural network for story segmentation. In Proc. of Interspeech.

Turing, A. M. (1950). Computing Machinery and Intelligence. In Mind, Oxford University Press, Mind Association (pp. 433–460).

Uchyigit, G. (2009). Semantically Enhanced Web Personalization. Web Mining Applications in E-commerce and E-services. Springer Berlin Heidelberg, 25–44.

Utiyama, M., & Isahara, H. (2001). A Statistical Model for Domain-independent Text Segmentation. In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (pp. 499–506).

Vassileva, J. (1996). A task-centered approach for user modeling in a hypermedia office documentation system. User Modeling and User-Adapted Interaction, 6(2), 185–223.

Verma, B., & Thakur, R. S. (2018). Sentiment Analysis Using Lexicon and Machine Learning-Based Approaches: A Survey. Proceedings of International Conference on Recent Advancement on Computer and Communication. Springer Singapore, 441–447.

Vinokourov, A., Cristianini, N., & Shawe-Taylor, J. (2003). Inferring a semantic representation of text via cross-language correlation analysis. In Advances in neural information processing systems (NIPS 2003) (pp. 1497–1504).

Viveros-Jiménez, F., Sánchez-Pereza, M. A., Gómez-Adorno, H., Posadas-Durán, J. P., Sidorov, G., & Gelbukh, A. (2018). Improving the Boilerpipe Algorithm for Boilerplate Removal in News Articles Using HTML Tree Structure. Computación y Sistemas, 22(2).

Vodolazova, T., Lloret, E., Mu, R., & Palomar, M. (2013). Extractive Text Summarization : Can We Use the Same Techniques for Any Text ? International Conference on Application of Natural Language to Information Systems. Springer Berlin Heidelberg, 164–175.

Vogels, T., Ganea, O.-E., & Eickhoff, C. (2018). Web2Text: Deep Structured Boilerplate Removal. Advances in Information Retrieval. Springer International Publishing, 167–179.

Vossen, P. (1998). A multilingual database with lexical semantic networks. Kluwer Academic Publishers.

Wang, L., Li, S., Lv, Y., & Houfeng, W. (2017). Learning to rank semantic coherence for topic segmentation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 1340–1344).

Wang, S. (2008). Ontology of learning objects repository for pedagogical knowledge sharing. Interdisciplinary Journal of E-Learning and Learning Objects, 4(1), 1–12.

Wang, Z., Zhang, J., & Huang, J. (2017). Multi-granularity hierarchical topic-based segmentation of structured, digital library resources. The Electronic Library, 35(1), 99–120.

Weal, M. J., Alani, H., Kim, S., Lewis, P. H., Millard, D. E., Sinclair, P. A. S., Shadbolt, N. R. (2007). Ontologies as facilitators for repurposing web documents. International Journal of Human-Computer Studies, 65(6), 537–562.

Weizenbaum, J. (1966). ELIZA–a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36–45.

Wiley, D. A. (2000). Learning Object Design and Sequencing Theory. PhD Thesis. Brigham Young University.

Willett, P. (2006). The Porter stemming algorithm then and now. Program, 40(3)

Wilson, C., & Scott, B. (2017). Adaptive systems in education: a review and conceptual unification. International Journal of Information and Learning Technology, 34(1), 2–19.

Witten, I., & Milne, D. (2008). An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA (pp. 25–30).

Wu, H., De Bra, P., Aerts, A., & Houben, G.-J. (2000). Adaptation control in adaptive hypermedia systems. In Adaptive Hypermedia and Adaptive Web-Based Systems (pp. 250–259).

Wu, Z., & Palmer, M. (1994). Verbs Semantics and Lexical Selection. In Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics (pp. 133–138).

Xu, C., Xie, L., & Xiao, X. (2017). A Bidirectional LSTM Approach with Word Embeddings for Sentence Boundary Detection. Journal of Signal Processing Systems, 1–13.

Xu, J., Gao, J., Toutanova, K., & Ney, H. (2008). Bayesian Semi-supervised Chinese Word Segmentation for Statistical Machine Translation. In Proceedings of the 22Nd International Conference on Computational Linguistics (pp. 1017–1024).

Yaari, Y. (1997). Segmentation of Expository Texts by Hierarchical Agglomerative Clustering. In Recent Advances in NLP (RANLP'97).

Yadav, V., & Bethard, S. (2018). A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 2145–2158).

Ye, N., Gauch, S., Wang, Q., & Luong, H. (2010). An Adaptive Ontology Based Hierarchical Browsing System for CiteSeerx. In 2010 Second International Conference on Knowledge and Systems Engineering (pp. 203–208).

Ye, N., Zhu, J., Zheng, Y., Ma, M., Wang, H., & Zhang, B. (2008). A Dynamic Programming Model for Text Segmentation Based on Min-Max Similarity. Information Retrieval Technology. Springer Berlin Heidelberg, 141–152.

Zeleny, J., Burget, R., & Zendulka, J. (2017). Box clustering segmentation: A new method for vision-based web page preprocessing. Information Processing & Management, 53(3), 735–750.

Zeng, D., Liu, K., Chen, Y., & Zhao, J. (2015). Distant supervision for relation extraction via piecewise convolutional neural networks. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 1753–1762).

Zhang, K., Sun, M., & Xue, P. (2010). A Local Generative Model for Chinese Word Segmentation. Information Retrieval Technology. Springer Berlin Heidelberg, 420–431.

Zhang, Y., Chan, W., & Jaitly, N. (2017). Very deep convolutional networks for end-to-end speech recognition. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 4845–4849).

Zhou, D., Goulding, J., Truran, M., & Brailsford, T. (2007). LLAMA: Automatic Hypertext Generation Utilizing Language Models. In Proceedings of the Eighteenth Conference on Hypertext and Hypermedia (pp. 77–80).

Zhou, D., Truran, M., Brailsford, T., Ashman, H., & Pourabdollah, A. (2008). Llama-b: Automatic Hyperlink Authoring in the Blogosphere. In Proceedings of the Nineteenth ACM Conference on Hypertext and Hypermedia (pp. 133–138).

Zouaq, A., Jovanovic, J., Joksimovic, S., & Gasevic, D. (2017). Linked Data for Learning Analytics: Potentials and Challenges. Handbok of Learning Analytics, 347–355.