Accepted Manuscript

Modelling text prediction systems in low- and high-inflected languages

Nestor Garay-Vitoria, Julio Abascal

PII: S0885-2308(09)00024-2
DOI: 10.1016/j.csl.2009.03.008
Reference: YCSLA 411

To appear in: Computer Speech and Language

Received Date: 11 October 2007
Revised Date: 18 December 2008
Accepted Date: 3 March 2009

Please cite this article as: Garay-Vitoria, N., Abascal, J., Modelling text prediction systems in low- and high-inflected languages, Computer Speech and Language (2009), doi: 10.1016/j.csl.2009.03.008

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
Modelling text prediction systems in low- and high-inflected languages
Nestor Garay-Vitoria, Julio Abascal
Laboratory of Human-Computer Interaction for Special Needs
University of the Basque Country
Manuel Lardizabal 1; E-20018 Donostia (Spain)
{nestor.garay, julio.abascal} @ehu.es
Tel: +34 943018000
Fax: +34 943015590

Abstract. Text prediction was initially proposed to help people with a low text composition speed to enhance their message composition. After the important advancements obtained in the last years, text prediction methods may nowadays benefit anyone trying to input text messages or commands, if they are adequately integrated within the user interface of the application. Diverse text prediction methods are based in different statistic and linguistic properties of natural languages. Hence, they are very dependent on the language concerned. In order to discuss general issues of text prediction it is necessary to propose abstract descriptions of the methods used. In this paper a number of models applied to text prediction are presented. Some of them are oriented to low-inflected languages while others are for high-inflected languages. All these models have been implemented and their results are compared. Presented models may be useful for future discussion. Finally, some comments related to the comparison of previously published results are also done.

Keywords: Text prediction; Anticipative interfaces; Prediction models; Communication speed enhancement; Prediction measures.

1. Introduction

Text prediction was initially designed to help people with a low text composition rate to enhance their message composition. This includes people with severe speech and motor disabilities such as cerebral palsy, hemiplegia, etc. Nowadays, text prediction methods, if adequately integrated within the user interface of an application, can benefit anyone trying to produce text messages or commands (Garay-Vitoria and Abascal, 2006). This is the case with the T9 keyboard (http://www.T9.com), which is a reduced keyboard commonly used for composing SMS messages in mobile phones. Text prediction can also be applied to other types of reduced keyboards (Abascal et al., 2004; Arnott and Jayed, 1992).

In the area of assistive communication, a predictor is a system which attempts to anticipate the next block of characters (letters, syllables, words, sentences, etc.) that the user wants to express. In general, prediction is based on the previously produced blocks (i.e. context information) (Garay and Abascal, 1994). The main aim of a predictor is to reduce the effort required and the message composition time. In order to reduce the effort, it is necessary to decrease the number of keystrokes needed for composing a message. To reduce the time needed, the predictor has to offer longer blocks (i.e. blocks composed of more characters) than the items contained in the selection set.

In other words, if the system is able to guess correctly, the number of keystrokes needed to write a sentence decreases and, apart from enhancing the speed of communication, the physical effort required to compose the messages is also reduced. In addition, the
prediction software may also correct spelling mistakes, reorder sentences and, in
general, enhance the quality of the composed messages (Magnuson, 1995; Zordell,
1990).
Prediction systems can be seen as intelligent agents that assist users in composing texts
(Garay-Vitoria and Abascal, 2006; Väyrynen, 2005). They capture user inputs to make
their guesses and produce outputs which can be incorporated into the applications used
to compose texts. They try to emulate user behaviour. Hence, the most advanced of
them have learning features, are able to make inferences, are adaptable and act
independently. Moreover, in certain cases they may converse with users, mainly to
make personal vocabulary adaptations.
Over the last few years several research teams have reported many results on text
prediction. They are based on diverse language features and present important
differences, including prediction algorithms, dictionary organisation, interfaces, etc. A
number of them are commercially available, while others have only reached the
prototype phase. Thus, it is difficult to compare their performance, as the measurements
used by the authors are based on heterogeneous parameters, not always clearly
described. A survey of text prediction systems can be found in Garay-Vitoria and
Prediction based on characters or n-grams only needs a table to store data on the blocks
and their frequencies. On the other hand, word prediction systems store the information
they require in dictionaries or lexicons (Garay-Vitoria, 2001). Predictors that treat units
longer than one word have larger storage requirements. For each sentence, they usually
have to store information on the context, block type, frequency, type of conversation,
etc. (Alm et al., 1992; Garay-Vitoria et al., 1995).
Most of the references on prediction analyse low-inflected languages such as English. In
this paper, we will also consider high-inflected languages such as German, Swedish,
Finnish, Basque, etc. A language is high-inflected when it is possible to produce a large
number of inflections starting from a single root or lemma. The root is the part
representing the most significant aspects of semantic content of the word and remains
unchanged in every inflection for the same semantic family. In contrast, we consider a
language as low-inflected when only a small number of inflections may be obtained by
starting from a single root (as in the cases of English, French or Spanish, for example)
(Garay-Vitoria, 2001; Garay et al., 2002).
In low-inflected languages, inflections are mainly derived from the number (and, in
some cases, the gender). For instance, the number may be singular or plural
(house/houses, spy/spies, etc.). In certain languages (for example, German), the gender
may be masculine, feminine or neuter.
When there are only a few variations of a word, it is possible to store all of them in the
dictionary. However, as inflected languages produce many inflections it may be difficult
to store all of them1. This is the main reason for the search for new prediction methods
in languages with a wide use of prefixes, infixes and suffixes. Work related to this field
can be found in Beck et al. (2004), Bertenstam and Hunnicutt (1995), Carlberger
Therefore, the possible savings are a function of the block size and the hit ratio of the
predictor (Garay and Abascal, 2004). In this paper, we only consider blocks composed

---

1 In the case of the Basque language, starting from a given root, 62 basic inflections may be obtained. Suffixes may
be recursively concatenated (as it is an agglutinated language) increasing the number of possible inflections. With a
two-level recursion, it has been estimated that a noun may reach 458,683 variations (Agirre et al., 1992). Prefixes
and infixes are also possible in Basque; however, in our studies on prediction, we have included them within the
roots, as their probabilities are not very significant (Garay et al., 2002).
of alphabetic characters (other symbols and pictograms are not taken into account); i.e. we focus on text prediction.

This paper is organised as follows: Section 2 presents models that can be applied to developing text prediction systems for low-inflected languages; Section 3 presents models for high-inflected languages; Section 4 then presents some results obtained with prediction systems developed by authors. Finally, Section 5 draws some conclusions and explores future trends.

2. Models of text prediction systems in a low-inflected language

In this section, we present the predictor models we have developed for the Spanish language, a low-inflected language according to the definition made in the previous section. We present the different methods of prediction, from the least complex to the most complex.

In Table 1 we present the initial basic definitions that we use in the models we are going to present.

2.1 Prediction using frequencies

Prediction using frequencies is the simplest method for making anticipations. It makes use of a dictionary which contains words and their frequencies. It only takes into account the current word; i.e. the word that is being composed by the user, character by character. The system forms a string with the characters that it receives. This string is the base for anticipating the word that the user is going to compose (see Table 1).

Taking $\alpha$, the initial character string, the predictor proposes the most frequent $n$ words that have not been previously proposed as the current word, in the same form they are stored in the dictionary. Then, the user may accept one of the proposals the system offers (if the word he/she wants to write is among them) or else the user continues composing the word. This process continues until the predictor guesses the word the user is composing and the user accepts it, or the user completely writes the word. Then, the system considers the next word and the process is repeated.

The dictionary for this system is adaptable. When the user accepts an option proposed by the system or he/she writes a word that is in the dictionary, the frequency of the word is incremented. When the word is not in the dictionary, it is included with a frequency of one, provided it is spelled correctly. In the case of acceptance, words are included in the text that is being composed, with a final blank space.

In this approach, words are seen as isolated entities; i.e. syntax and context information is not included, and only frequencies are taken into account.

The mode of operation of this approach is shown in Table 2. This table and the procedure associated with it may be used, with some changes, in the other approaches that will be presented later.

In this mode of operation, it is assumed that the acceptance is done by writing a number. To adapt the system, it is only necessary to update the frequencies in the dictionary each time they are used. Including new words in the dictionary is automatic and there is no
need for user collaboration. In subsequent approaches that are presented in this paper, when the words in dictionary are associated with a greater amount of information, the inclusion of new words is a problem that needs to be resolved. We can conclude by saying that in prediction using frequencies the system always has the information that is needed for its correct and optimal functioning. Algorithm 1 is the one associated with this prediction.

As can be seen, deleting a previously written character is equivalent to recuperating and making valid the immediately previously shown proposals (Garay-Vitoria, 2001). Conditional acceptance happens when the correct lemma of the word is proposed but with an incorrect word ending: for instance, the gender or number of the word is not correct. The user can accept the lemma and then change it in order to correct this. It is also possible to make corrections using syntax correctors that run after the prediction software.

This approach has been used as the baseline for prediction. Furthermore, it has been used to make comparisons with the other approaches for the Spanish language.

The main advantage of this approach is that it is language independent. There is only the need to have a dictionary with words and frequencies. It is possible to make predictions without considering more language-specific information. This is the reason why this approach was the first to be used for Basque prediction, which is presented in Section 3 of this paper.

### 2.2 Syntactic prediction

While in prediction using frequencies words are seen as isolated entities, in syntactic approaches they are seen as parts of sentences. Morphosyntactic information is also taken into account to enhance the results that are achieved by prediction using frequencies.

A word in the dictionary is associated with its syntactic category and its frequency of appearance. For words with ambiguous categories, diverse strategies are discussed by Garay-Vitoria and colleagues in Garay-Vitoria and Abascal (1997a). The most common strategies are either to create a new category for each possible ambiguity, or to associate a list of categories with each word. Later the predictive system has to cope with ambiguous categories of the previous composed words when composing a new word in the text. These categories have to be taken into account while making predictions.

When more than one category is applicable, each word stored in the dictionary is associated to the list of possible categories. The system computes all the categories in the list equally distributing the probability among them. If new words are included in a system that uses morphosyntactic information, it is necessary to include this information (and sometimes an initial frequency) with the word. The ideal solution would be a system which automatically includes this information in the lexicon. As morphosyntactic information is not always evident, the system may include erroneous categories, which would make subsequent predictions difficult. Other strategies allow the system to ask the user about the characteristics of new words as they appear. If the user does not know the morphosyntactic characteristics of a word, similar problems arise. In addition, this strategy decreases the speed of message composition. To avoid this, new words are included with an empty category and the user is later prompted to complete the necessary information.
2.2.1 Syntactic prediction using probability tables

In this approach, two statistical data types are used: the probability of appearance of each word and the relative probability of appearance of every syntactic category after the previous syntactic category. These systems suggest the words with the most probable categories in the current position of the sentence. The results are usually better than the ones obtained with the previously shown strategy.

A two-dimensional table stores the probabilities of appearance of the categories after each preceding category. Its dimensions are set before the predictive system is built. These systems are personalised by updating the probabilities in the table and the frequencies in the lexicon. If new words are added, some of the strategies discussed above have to be applied. Several systems implement this strategy (Garay and Abascal, 1994; Swiffin et al., 1987a).

The system also takes into account whether the current word is the first one in the sentence.

Table 3 includes the new definitions for this approach (where several definitions are substituted for some of the previous ones, such as, for example, Word\textsuperscript{n} (α)). The mode of operation is the same as the one shown in Table 2, but taking into account the new Word\textsuperscript{n} (α). Algorithm 2 shows how this approach works. In this algorithm, invalidation and revalidation of proposals are not mentioned as they are the same as in prediction using frequencies.

\begin{table}
\centering
\caption{New definitions for \ldots}
\end{table}

\begin{algorithm}
\caption{Algorithm for \ldots}
\end{algorithm}

In this approach, a category of ‘Unknown’ is initially given to new words when they appear. Later on, the morphosyntactic information they require has to be completed in special sessions.

This approach only takes into account the current word and previous word and their categories. It can be adjusted to take into account more than one previous word and transition probabilities.

2.2.2 Syntactic prediction using grammars

Sentences are analysed using grammars and by applying Natural Language Processing techniques from the Artificial Intelligence field, in order to obtain the categories which have the highest probability of appearance. Methods for analysing the sentences may be top-down (Van Dyke, 1991) or bottom-up (Garay-Vitoria and Abascal, 1994). Each natural language has a set of syntactic rules. These rules usually have the following structure:

\begin{equation}
\text{LEFT} \leftarrow [\text{RIGHT}]^+. \quad (1)
\end{equation}

This means that the category on the left of the rule may be decomposed into the sequence of categories on the right, in the order in which they appear. At least one category has to appear on the right and all the categories are defined in the system. For example, if NP represents a Noun Phrase, PP a Prepositional Phrase, Noun a noun and Prep a preposition, the following rules apply:

\begin{equation}
\text{NP} \leftarrow \text{Noun PP}. \quad (2)
\end{equation}
may be expanded to give following:

\[ \text{NP} \leftarrow \text{Noun Prep NP.} \]  

Therefore, recursive application of the rules is possible. In addition, it is possible to define a number of grammatical concordances (such as the grammatical number), among the categories on the right of a rule. In this way, the predictor can offer proposals based on their most appropriate morphological characteristics.

The dictionary used is very similar to the one in the previous approach. However, to make grammatical concordances possible, it requires the addition of morphological information. These systems have a higher computational complexity than the previous ones, mainly due to the fact that they take the entire beginning of the sentence into account (while the previous systems only consider, at most, the last entirely composed word). These types of systems may be adapted by updating the word probabilities and weightings of the syntactic rules (Hunnicutt, 1989; Wood and Lewis, 1993). To add new words to the dictionary, the solutions mentioned in the beginning of section 2.2 may also be used here.

While in other applications (such as morphosyntactic correctors) there is a need for the complete grammar of the language, it is not necessary here. It is possible to have a number of the most probable rules that will be computationally affordable in order for the system to make proposals in time (between two consecutive user keystrokes).

In our systems, sentences are parsed using the chart bottom-up method (Allen, 1995). The rules that are partially completed at this point of the sentence are used in order to guess the most probable categories. Table 4 and Algorithm 3 show the definitions and the mode of operation of this approach. As it can be seen in Table 4, the probability of a rule is directly proportional to its frequency of occurrence.

3. Models of text prediction systems in a high-inflected language
In this section, we present the models of the predictors we have developed for the Basque language, a high-inflected language according to the classification made in Section 1. As in Section 2, we present several prediction methods, from the simplest one to the most complex one.

As they take both lemmas and affixes into account, in general prediction is divided into several steps. In our methods, usually two steps are used: one for lemmas and the other for suffixes, because the other affixes are not very frequent in the Basque language. In general, two dictionaries are used: one for lemmas and the other for suffixes (Garay et al., 2002).

3.1 Word prediction using frequencies

As was mentioned in subsection 2.1, word prediction using frequencies is language independent. There is only the need to have a dictionary containing words and their frequencies. Therefore, the first approach taken for the Basque language is the same as the first one used for Spanish. Its mode of operation and basic definitions are those shown in Tables 1 and 2.

The main objective in taking this approach is to use it as the baseline for prediction and for making comparisons with the other approaches for the Basque language, similar to the approach in subsection 2.1.

3.2 Morpheme prediction using frequencies

Only taking statistical data into account, this approach has two dictionaries: one with roots and their frequencies and the other with suffixes and their frequencies. From this approach onwards, in order to make predictions both user and system have to clearly distinguish whether roots or suffixes are being considered. One way to clearly distinguish if the system is working with roots is by including some dots in the interface at the end of lemmas (for example, “know…”); similarly, the system will be working with suffixes if proposals are preceded by dots (for instance, “…ing”). The cognitive interface must also offer the feature of explicitly changing from root prediction to suffix prediction and vice versa. This change is made implicitly when proposals are accepted or when the user finishes composing a unit (either a root or a suffix), with perhaps a final space or a similar character.

This system starts by proposing the most frequent roots that begin with the string the user has written so far, until the root is guessed or entirely written; it then proposes the suffixes until the word is guessed or completed. Next, it starts again with roots and so forth. Adaptations are made by updating frequencies.

The algorithms related to this approach are presented in Table 5. As can be seen, they are similar to the one presented in Table 2. However, it must now be clearly indicated when it is treating roots and when suffixes. Furthermore, more keystrokes are now needed for completing each word, because of the need for writing characters to make explicit changes from root treatment to suffix treatment and vice versa. In the previous approach, it is possible to complete a word with only one acceptance whereas in this one at least two acceptances are needed: one for roots and the other for suffixes.

This system has to cope with irregular cases to correctly complete words composed of a given root and suffix.

Verbs are a special case, as found in this study, because their suffixes are different in relation to the rest of the words. In order to achieve a better treatment of this special
3.3 Morpheme prediction using probability tables taking categories into account

This also distinguishes between roots and suffixes, and adds syntactical information to lexicons. Moreover, there are several two-dimensional tables that store the relative probabilities of suffix categories appearing after root categories and vice versa, for making predictions.

The categories of the roots are usual in other languages (nouns, verbs, adjectives, and so on) whereas suffix categories refer to the case that can be applied to the root (for instance, dative, ergative, absolutive and similar cases can be applied to nouns or adjectives).

In this system, the most probable roots starting with the string already written and whose categories are the most probable after the last composed suffix, are then proposed. After composing the root, the suffixes are then treated, and so on. Adaptation is performed by updating the frequencies of the lexicons and the entries in the tables used (Garay-Vitoria, 2001; Garay et al., 2002).

The system has two probability tables: one for promoting root categories (in the case where roots are being treated) and the other to promote suffix categories, when suffixes are being considered. The dimensions of the tables are quite similar: the first one will have around M*N entries, M being the number of root categories and N the number of suffix categories, and the second one will have around N*M entries.

The algorithms for this approach are those shown in Table 5, but probability tables have to be taken into account when calculating Rootn(α) and Suffixn(α) and also when doing the updates.

However, there are various problems to solve, as there are ambiguous morphemes and new roots and suffixes that may appear if adapted to the user's vocabulary. Possible ways of solving these problems may be found in the literature (Garay et al., 2002): systems that typically try to guess the category; others that directly ask the user for the missing information; or some systems that give a special category (related to the new roots or suffixes), in order for the information to be completed when the user is ready to do so (perhaps by a specific user or in a specific session).

3.4 Morpheme prediction using syntactic grammars

In the next two cases, some grammars are defined using rules that show how some composed categories may be decomposed into a succession of categories (which can be simple or composed). Each rule has a weighting that refers to its probability of appearance, directly proportional to its frequency of occurrence, as in the case of the low-inflected language. There is a probability table of the syntactic categories of the roots to start sentences that is used for prediction (Garay-Vitoria, 2001; Garay et al., 2002), similar to the table presented in Subsection 2.2.2. The sentences are parsed using the chart bottom-up method.

In this system, the dictionaries continue storing syntactic information. The system offers the most probable roots with the appropriate start whose categories are most likely to appear after the categories at the beginning of the sentence; then, the suffixes are proposed in a similar way and the process is repeated again and again (Garay et al., 2002).

With the first word of a sentence, the most probable roots are proposed based on the probability table. As seen, the entire start of the sentence is considered. Therefore, the
computational complexity of this approach is higher than the previous ones (Garay et al., 2002).

In the adaptation, there is a need to update the frequencies of the dictionaries, the weightings of the rules and the entries in the probability table. Once again, the algorithms for this approach are those shown in Table 5, but partially completed rules have to be taken into account when calculating Root\(\alpha\) and Suffix\(\alpha\) and also when doing the updates.

To alleviate the cognitive complexity associated with this approach and to try to enhance the results achieved, a new approach will be to consider word prediction again.

### 3.5 Morpheme-based prediction using syntactic grammars with acceptance of entire words

In our studies, we have found it is quite common in Basque for a root to act as an entire word (that is to say, very often no suffix is added to a root). This approach uses this feature as a basis for enhancing the results achieved by the previous approaches (Garay et al., 2002).

The lexicons, the rules and the probability table are the same as in the previous approach. It tries to guess the entire word if possible; if not, the roots are tried and then the suffixes, as in the case of the previous approach. Thus, as can be seen, the entire beginning of the sentence is once again taken into account.

The same idea could be applied to enhance the approaches presented in Subsection 3.2 and 3.3, but as it was found that the approach presented in Subsection 3.4 achieved the best results so far, the idea was only applied to this approach. The acceptance protocol is changed to accept roots either as entire words or as roots.

The adaptation is also done in the same way as in the previous approach. In order to distinguish whether a root is accepted as a word or a mere root, the acceptance protocol is changed to make total and partial acceptances possible.

The mode of operation of this approach is shown in Table 6. Root\(\alpha\) and Suffix\(\alpha\) are calculated taking into account the entire beginning of the sentence, keeping in mind the partially completed rules of the grammar. In the beginning of the sentence, Root\(\alpha\) is calculated using the starting probability table. To distinguish between total and partial root acceptances, we define Number(c) as expressing a root that is accepted as part of the current word and NUMBER(c) as expressing a root that is accepted as the current word. In the case of suffixes, number(c) expresses a suffix that is accepted as the ending of the current word. ROOTEND is a character that indicates the writing of the current lemma is finished and that next the system will be making a suffix prediction.

| Place Table 6 around here. |

### 4. Results and discussion

There are two main methods of evaluating the prediction systems presented in the literature. The first one, *emulation*, feeds the predictor with several trial texts, emulating human behaviour. The second method, *human testing*, employs a sample of real users who try to write specific texts. In emulation mode, a program tries to write trial texts character by character in the same way they would be written by an ideal human. The emulator program receives proposals from the predictor and only accepts those that coincide with the item being composed at the moment. In this way items are written by the emulator one by one with the help of the predictor.
In our studies, items are Spanish words separated by a blank character, while in most approaches for Basque language items should be roots or suffixes and sometimes delimiters. The emulator must know whether root or suffix prediction is being made in order to use the correct predictor engine.

Emulation always produces the best possible results in terms of time savings, as the program “types” at the highest possible speed. However, this method has the disadvantage that it does not take human features, such as fatigue or error making, into account. Nevertheless, this method is relatively objective and it is useful in taking a number of design decisions. Many authors show the results of using evaluation methods (Carlberger et al., 1997; Garay and Abascal, 1994; Higginbotham, 1992; Hunnicutt, 1987; Swiffin et al., 1987b; Wood, 1996).

On the other hand, empirical evaluation with users allows more reliable data to be obtained with regard to the message composition speed, as the test includes human factors such as error rate, fatigue, learning time, etc. Moreover, it is possible to measure how the system is used (for example, if the user reads all the proposals made by the system before making a selection), whether the system and its interface copes with the user's objectives (i.e. if it is easy to learn and use, if a faster message composition is achieved, if a better orthography is achieved, and so on). Data related to the user's preferences may also be obtained. Therefore, several authors prefer to perform tests with real users (Carlberger et al., 1997; Koester and Levine, 1994; Newell et al., 1992; Venkatagiri, 1993).

In our opinion, the second method is the better one for obtaining a validation of the design under real conditions; however, it presents the difficulty of finding a set of people who represent the entire population of possible users of the system. Moreover, the confidentiality of the results may also be a problem (Copestake and Flickinger, 1998) and there are authors who suggest this type of study may be very limited and should be confined to individual case studies (Digiovanni, 1996). This is the reason why, in the design phase (testing and selecting the different characteristics of the system), predictors are frequently evaluated by simulations in order to compare their characteristics in an objective way.

In the case of the Spanish language, the dictionary was created by means of written texts that were published in modern press in addition with words that are usual for people with disabilities, as the National Institute for Migration and Social Services (also known as IMSERSEO) made us known (Garay-Vitoria, 2001). This combination mixes recent vocabulary with the vocabulary that is demanded by the most probable users of this type of systems. In the case of the Basque language, dictionaries have been created by using several modern written material (including conversation transcriptions) that were facilitated by the most representative research group in Computational Linguistics in Basque language (IXA Taldea group) (Garay-Vitoria, 2001). In this particular case it was nearly impossible to incorporate their usual vocabulary to the system due to the low number of speakers with disabilities in this language and the subsequent lack of studies about their lexicon.

4.1 Results achieved for the Spanish language
The anticipation methods modelled for low-inflected languages have been implemented for the Spanish language. These predictors display 1, 5 or 10 options trying to guess the word that is going to be written. As expected, increasing the number of options shown, improves the results. However, in this case, it will be slower for a real user, reading all of them, to make a decision about accepting or rejecting them. Moreover, increasing this
number of options makes it difficult to display them in the interface (Garay-Vitoria and Abascal, 2004).
The dictionary that has been used is composed of 5,040 words, with and without morpho-syntactic categorization. These words have been selected from some written corpora, also taking into account some sentences written by real users. The used testing text is a different set of sentences written by real users. This text has 2,444 characters. Other texts and results can be found in (Garay and Abascal, 1994; Garay-Vitoria, 2001; Garay-Vitoria and Abascal, 1994; Garay-Vitoria and Abascal, 1997a).
The grammar defined is composed of the relevant rules found in several corpora. To do that each word in these corpora were substituted by its morpho-syntactic category. Later the most frequent structures were selected (taking into account the Spanish grammar) and a number of rules, each one with its own frequency, were created. Finally, only the 64 most frequent rules were selected, because they were the most significant ones.

Figures 1 and 2 show the results achieved with the three methods presented. In all cases, prediction using frequencies achieved the worst results in terms of keystroke savings and hit ratio.

Keystroke savings (KS) are defined in (5).

\[
KS = \frac{\text{total}\#\ _\ of \ _\ keystrokes - \#\ of \ _\ written\ _\ keystrokes}{\text{total}\#\ _\ of \ _\ keystrokes}
\]  

The hit ratio (HR) in word prediction is the probability of guessing a word. It is defined as in (6).

\[
HR = \frac{\#\ _\ of \ _\ times \ a \ _\ word \ _\ is \ _\ guessed}{\#\ _\ of \ _\ written\ _\ words}
\]

Prediction using probability tables achieves similar results to prediction using grammars in terms of keystroke savings, but it has a lower hit ratio. The second approach enhances the keystroke savings of the first one by between 2.83% and 4.60% in absolute values (8.56%-11.32% in relative values). The third one enhances the results of the second one by between 0.07% and 1.03% in absolute values (0.24%-2.18% in relative values).

4.2 Results achieved for the Basque language

In every approach, the results that are achieved when the system offers 1, 5 or 10 proposals to the user are studied.

The word prediction approach is associated with a lexicon that has been created starting from some Basque corpora. This lexicon has 8,969 entries, each composed of a word and its frequency. The lexicon of the rest of the approaches is also taken from those corpora, decomposing words into their roots and suffixes. In concrete terms, they have 4,771 roots and 233 suffixes (44.21% items less than in the first approach).

Similarly to the Spanish grammar, the Basque grammar defined is composed of the relevant rules found in several corpora. Words in these corpora were substituted by their morpho-syntactic categories and the emergent category lists were extracted. Later the most frequent lists were analysed comparing them with Basque grammar and several
rules were defined, each one with its own frequency. Finally, the most frequent rules were taken into account. The amount of considered rules was 111.

In this paper, we show the results achieved with a testing text selected from a set of several testing texts used in authors’ previous studies. All the texts were selected from data that was not used to build the lexicon, in order to evaluate results that can be obtained in a not previously expected situation (Garay-Vitoria, 2001; Garay et al., 2002). As there is a need to distinguish roots and suffixes in four of the approaches (it is not necessary in the first), we have two versions of the text: one with delimiters of roots and suffixes (for the last four approaches) and another without the delimiters (for the first), as shown in Table 7. Thus, depending on the version used, the text has 2,691 or 2,974 characters. The text is a technical report related to computer science and the Internet.

In Figure 3, we show the keystroke savings, in percent, obtained using the different approaches with the aforementioned text.

The main qualitative enhancements appear when the morphemes are treated separately (instead of treating the words) and when a root can be accepted as a whole word (instead of not giving this option). The second approach enhances the results of the first by between 5.83% and 11.08% in absolute values (28.24%-35.89% in relative values). The third approach presented enhances the results of the second by 1.55%-2.66% in absolute values (4.80%-6.50% in relative values). The fourth is better than the third by 0.27%-1.11% absolute (0.62%-3.15% relative) and the last enhances the fourth by 2.63%-3.13% in absolute terms (6.87%-9.16% relative enhancement).

Figure 4 shows the hit ratios achieved. In the first approach (6) formula is applied. In the other approaches, hit ratio can be more precisely defined as the fraction of terms where some help was given by the system (i.e. either the root or the suffix was predicted). In approaches 2nd to 4th -where roots and suffixes are distinguished- hit ratio is calculated as in (9), with HRR being the hit ratio for roots (7) and HRS the hit ratio for suffixes (8). Since a word must be composed in these approaches by a root and a suffix, HR will be calculated by adding HRR and HRS and then dividing the addition by 2. In the last approach, where HRW is the hit ratio for roots that work as words as in (10), P the frequency of occurrence that there is no suffix added to a root in the used text (it is calculated dividing the times no suffix is added to a root in the text by the number of words in the text, 1>=P>=0) and HRR and HRS are as previously defined, we define HR as in (11). In this case, a word can be completely guessed when no suffix is added to the root or it can be partially guessed in the other cases, as in (9) formula.

\[
HRR = \frac{\# \_ of \_ times \_ a \_ root \_ is \_ guessed}{\# \_ of \_ written \_ roots} \quad (7)
\]

\[
HRS = \frac{\# \_ of \_ times \_ a \_ suffix \_ is \_ guessed}{\# \_ of \_ written \_ suffixes} \quad (8)
\]

\[
HR = \frac{HRR + HRS}{2} \quad (9)
\]
HRW = \frac{\# \text{ of } \text{times } \text{a } \text{root } \text{is } \text{guessed } \text{as } \text{a } \text{complete } \text{word}}{\# \text{ of } \text{written } \text{words}} \quad (10)

HR = HRW \times P + \left( \frac{HRR + HRS}{2} \right) \times (1 - P) \quad (11)

In general, the last approach presented achieves the best results, with a hit rate of between 12.75% and 15.23% higher than the next best (generally the fourth one). This enhancement with respect to the rest is mainly due to the fact that when accepting a root as a word, there is an implicit acceptance of its null suffix, and this type of suffix is otherwise impossible to guess (at least in the approaches that distinguish roots and suffixes). The approaches that work with morphemes are all similar to one another. With one proposal, the word prediction presents a hit ratio similar to the morphemes and frequencies approach; however, in the rest of the cases it is the worst.

Place Figure 3 around here.

Place Figure 4 around here.

Generally, the anticipatory methods found in the literature express their results in terms of the savings they achieve. For example, most of the anticipators have been evaluated showing the keystroke savings they obtain (Garay and Abascal, 1994; Garay-Vitoria and Abascal, 1994; Hunnicutt, 1987; Hunnicutt, 1989; Swiffin et al., 1987a; Swiffin et al., 1987b). There are others which show the message composition time saved, which is directly associated with the keystroke savings (Copestake, 1997; Koester and Levine, 1994). The hit ratio may also be shown (Garay-Vitoria, 2001), as previously noted.

A rigorous comparison of the different methods appearing in the literature would require a standard workbench to establish a corpus from where to obtain the dictionaries, the trial texts to write and the type of measurements to be made (Garay-Vitoria, 2001). As this workbench does currently not exist, the results presented by several authors are very heterogeneous and practically impossible to reproduce (as the description of the algorithms used is usually superficial). Therefore, the comparisons made between them are purely estimative or informative and do not have evidential value as they are not verifiable, and they cannot be used to determine whether one approach is better than another.

There is a need to decide on the criteria necessary to establish a standardised workbench for making comparisons. As text prediction depends on the language used, a standardised workbench for each language would be a very good starting point for making competitions and comparisons between the text prediction systems developed for that language.

5. Conclusions

Within Augmentative and Alternative Communication Systems, text prediction techniques have often been designed with the aim of increasing the composition speed. These techniques were initially aimed at people with severe motor and oral disabilities, but non-disabled people can also use them when composing messages, as they can be
viewed as intelligent agents that assist users in text composition. In fact, today it is usual to use a prediction system while composing SMS messages. Several authors have proposed very different predictive techniques and strategies that are difficult to compare. The most advanced predictors have learning features, are able to make inferences, are adaptable and act independently. They may also interact with the user in some cases, primarily in adapting to the user’s lexicon.

In this paper a number of models that can be applied to text prediction have been presented. Due to the influence of the considered language, some of them have been developed for low-inflected languages and others for high-inflected languages. Every model presented has been implemented and the results achieved have been shown. Even if each of these models has been applied to a specific language, these authors think that they may also be applicable to other languages having similar linguistic structure. It is hoped that the models presented in this work will be used as a reference point for future discussion.

In both languages it has been found that increasing the number of proposals given by predictors enhances Keystroke Savings and Hit Ratio. Moreover, including morphosyntactical information also increases Keystroke Savings and Hit Ratio. For the Spanish text, results are up to 52.18% for Keystroke Savings and 96.80% for Hit Ratio, while for the Basque text Keystroke Savings is up to 48.08% and Hit Ratio, up to 83.44%. Anyway, interface issues have to be taken into account in order to properly adjust to user characteristics (Garay-Vitoria and Abascal, 2004).

It may also be of benefit to develop future models and systems for making iconic prediction, as there are people who express themselves better and languages that are better expressed using icons. These systems will also need a specific workbench in order to make proper comparisons.

References


Garay N, Abascal J. A comparison of prediction techniques to enhance the communication rate. LNCS 2004; 3196: 400-417.


Figure captions

Figure 1. Results achieved for the Spanish language with the methods presented.

Figure 2. Comparison between hit ratios achieved for Spanish approaches.

Figure 3. Keystroke savings for the Basque language with the different approaches.

Figure 4. Hit ratios achieved for the Basque language.
Algorithms

Algorithm 1. Mode of operation of the predictor using frequencies.

```plaintext
Open dictionary;
Read(c);
while No stop(c) do
    repeat
        if Delete character(c) then
            Delete last written character;
            Resend previously shown last proposals;
            Make valid again previously shown last proposals;
        else
            Make invalid last shown proposals;
        end if;
        Read frequencies of words beginning with new string;
        Select and show most frequent ones;
    Read(c);
    until Proposal accepted or No one word with this beginning or
    Word completely written;
    if No one word with this beginning then
        Give an indication about this;
        repeat
            Read(c);
        until Word completely written;
        Store word in dictionary;
    else
        if Proposal accepted then
            Update frequencies in dictionary;
            if User wants to change word ending then
                Change ending;
            end if;
        else
            if Word is in dictionary then
                Update frequencies in dictionary;
            else
                Store word in dictionary;
            end if;
        end if;
    end if;
end while;
Save updates in dictionary and close it;
```
Algorithm 2. Algorithm for syntactic prediction using probability tables.

Open files of dictionary and probability table;
Initialise table of probabilities of each category after the others;
Initialise category of last word;
Read(c);
repeat
  repeat
    Receive from dictionary words that have current beginning;
    Combine their frequencies with probabilities of categories;
    Propose words with highest probabilities;
    Read(c);
    until Guessed or No one word with this beginning or
    Completely written;
    if Guessed then
      Increment probability of next category;
      Increment frequency of the word;
    else
      if No one word with this beginning then
        Receive ending of the word;
        Include new word with Unknown category;
        Increment probability of next category;
      else
        if Word was in dictionary then
          Increment probability of next category;
          Increment frequency of the word;
        else
          Include new word with Unknown category;
          Increment probability of next category;
        end if;
      end if;
    end if;
  end if;
Read(c);
until End of session;
Close files saving changes;
Algorithm 3. Algorithm for syntactic prediction using grammars.

```
Open files and initialise tables and rules;
Read(c);
while Not end of session do
    repeat
        Use probability tables of categories to start sentence;
        Propose words with high frequency and category probability;
        Read(c);
        until Guessed or There are no words with current beginning or
        Word completely written;
        if There are no words with current beginning then
            Receive word ending;
        end if;
        Update probabilities and weightings;
        Generate a list of partially completed rules;
        Read(c);
        while Not end of sentence do
            Initialise weighting table;
            Calculate weightings for next categories;
            repeat
                Promote words of those categories;
                Propose words with appropriate gender and number;
                Read(c);
                until Guessed or There are no words with current beginning
                or Word completely written;
                if There are no words with current beginning then
                    Receive word ending;
                end if;
                Update probabilities and weightings;
                Generate new partially completed rules and modify existing ones;
                Read(c);
            end while;
            Read(c);
        end while;
    end while;
Close files with vocabulary adaptations;
```
Table 1. Set of basic definitions

- Let us denote the base alphabet as 
  \( A = \{ \text{`a', `b', ..., `z'} \} \)
- Let us denote a character string as 
  \( h \in A^* / \text{long}(h)=n \)
- Let us denote a reference text as 
  \( T \subseteq (A^*)^* \)
  defined as an array of words of the language
- Let us denote the set of elements within T as \( T_e \)
- Let us denote the reference dictionary where all the different words of T are stored as 
  \( D; \) it is defined as a prefixed finite subset of \( A \)
  \( \varnothing \text{fin}(A^*) / D \subseteq T_e \)
- We define frequency application as:
  \[ \text{frequency: } D \rightarrow \mathbb{R} / \text{frequency}(h) = \left\lfloor \frac{\sum_{x \in T \land x = h} 1}{\text{long}(T)} \right\rfloor \]
- Let us denote the dictionary with frequencies as 
  \( F = \{(h, f) / h \in D \land f = \text{frequency}(h)\} \)
- We define \( \prec \) as the order “be the initial segment” of \( A^* \)
  \( h \prec h' \) as long as \( \forall i (1 \leq i \leq \text{long}(h) \rightarrow h_i' = h_i) \)
- Given \( \alpha \), the string written by the user 
  \( \alpha \in A^* \)
  we define the set \( \text{Word} \) as the set of words in \( D \) that start with \( \alpha \):
  \( \text{Word}(\alpha) = \{h / h \in D \land \alpha \prec h\} \)
- Given \( \alpha \), the written string, and h, a word, according to Bayes’ theorem, the probability of h conditioned to \( \alpha \) may be calculated with the formula:
  \[ P(h | \alpha) = \frac{P(h) \ast P(\alpha | h)}{P(\alpha)} \]
  Given that \( P(\alpha) \) is common to all strings and that \( P(\alpha|h) \) is 1 or 0 (depending on whether \( \alpha \) is or is not the starting of h, respectively), \( P(h|\alpha) \) and \( P(h) \) are proportional (i.e. \( P(h|\alpha) \propto P(h) \)) and, instead of \( P(h|\alpha) \), \( P(h) \) may be used.
- We define \( \text{Word}^n(\alpha) \) as any set that satisfies the following conditions:
  1. \( \text{Word}^n(\alpha) = \{h/h \in D \land \alpha \prec h \land \forall h' \in \text{Word}(\alpha) \land \text{Word}^n(\alpha) \} \)
  2. \( \text{Card}(\text{Word}^n(\alpha)) \leq n \)
  for a pre-established given \( n \in \mathbb{N} \).
- Therefore, \( \text{Word}^n(\alpha) \) is a subset of \( D \) and its elements are the up to n most frequent words that start with the string \( \alpha \).
Table 2. Mode of operation of prediction using frequencies.

<table>
<thead>
<tr>
<th>n, D and F are established</th>
</tr>
</thead>
<tbody>
<tr>
<td>User is modelled as a generator of a finite character string</td>
</tr>
<tr>
<td>Objective: to guess the word the user is going to write</td>
</tr>
<tr>
<td>The system works in this way after a blank space is written:</td>
</tr>
</tbody>
</table>

```
Read(c); α:=<>;
while c∈A do
  α:= α · c;
  Show(Word^n (α));
  Read(c);
  if Number(c) and 1≤c≤Card(Word^n (α)) then
    Write_corresponding_suffix(c, α, Word^n (α)); /*complete word*/
  end if;
end while;
Make_updates;
```
Table 3. New definitions for syntactic prediction using probability tables.

- Let us denote the set of syntactic categories of the language as $S$.
- We define cat application as:
  $\text{cat}: \mathbb{D} \rightarrow S$ / $\text{cat}(h) = s$
- Let us denote the matrix that stores the absolute probabilities of a category appearing after the other categories as $M$:
  $\text{Dim}(M) = \text{Card}(S) \times \text{Card}(S)$
- We define index application as:
  $\text{index}: S \rightarrow \mathbb{N}$ / $\text{index}(s) = n \land 1 \leq n \leq \text{Card}(S)$
- Let us denote as last (last $\in \mathbb{D}$) the last word written by the user
- Now, we define prob application as:
  $\text{prob}: S \rightarrow \mathbb{R}$ / $\text{prob}(s) = \sum_{i,j} M[i, j]$ for $i = \text{index}(\text{last})$ and $j = \text{index}(s)$.
- Let us denote combi application:
  $\text{combi}: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
  any function that grows with the growth of its parameters.
- We define $k$ as the context of the word $h$ (in this case, the category of the last word), then:
  $P(h|\alpha k) = P(h|\alpha) \times P(h|k) \propto P(h) \times P(h|k)$
  As shown in Table 1, $P(h) \times P(h|k)$ can be used instead of $P(h|\alpha k)$.
  This is the reason why $\text{combi}$ is usually the product.
- We now define $\text{Word}^n(\alpha)$ as any set that satisfies the following conditions:
  1. $\text{Word}^n(\alpha) = \{h | h \in \mathbb{D} \land \alpha \prec h \land s = \text{cat}(h) \land \forall h' \in \text{Word}(\alpha) \setminus \text{Word}^n(\alpha) / s' = \text{cat}(h') \rightarrow \text{combi}(\text{frequency}(h'), \text{prob}(s')) \leq \text{combi}(\text{frequency}(h), \text{prob}(s))\}$
  2. $\text{Card}(\text{Word}^n(\alpha)) \leq n$
  For a pre-established given $n \in \mathbb{N}$.
- Then, $\text{Word}^n(\alpha)$ is a subset of $\mathbb{D}$ and its elements are the up to $n$ most frequent words that start with $\alpha$ whose syntactic category is the most probable one after the category of the last composed word.
Table 4. New definitions for syntactic prediction using grammars.

- Let us denote the set of the syntactic rules of the language as E.
- We define application weight as:
  \[ \text{weight} : E \rightarrow \mathbb{N} / \text{weight}(e)=n \]
  (i.e. n is the weighting of rule e)
- We define \( E_p \) (\( E_p \subset E \)) as the set of partially completed rules while composing the current sentence.
- We define next application as:
  \[ \text{next} : E_p \rightarrow S / \text{next}(ep)=s \]
  (category s is the next one to continue completing ep rule)
- We define prob application as:
  \[ \text{prob} : S \rightarrow R / \text{prob}(s) = \frac{\sum_{ep \in E_p : \text{next}(ep)=s} \text{weight}(ep)}{\sum_{e \in E} \text{weight}(e)} \]
- The context of word h is composed of the syntactic categories of the previous words in the sentence.
- Now we define \( \text{Word}^n(\alpha) \) as any set that satisfies the following conditions:
  1. \( \text{Word}^n(\alpha) = \{ h / h \in D \land \alpha \prec h \land s = \text{cat}(h) \land \forall h' \in \text{Word}(\alpha) \setminus \text{Word}^n(\alpha) / s' = \text{cat}(h') \rightarrow \text{combi}(\text{frequency}(h'),\text{prob}(s')) \leq \text{combi}(\text{frequency}(h),\text{prob}(s)) \} \)
  2. \( \text{Card}(\text{Word}^n(\alpha)) \leq n \)
  for a previous given \( n \in \mathbb{N} \).
- Therefore, \( \text{Word}^n(\alpha) \) is a subset of D and its elements are the up to n most frequent words that start with \( \alpha \) whose syntactic category is the most probable one after the categories of the words in the beginning of the sentence.
Table 5. Mode of operation of morpheme anticipation using frequencies

- User is modelled as a generator of a finite character string
- Objective: to guess the word (root + suffix) the user is going to write
- We include new definitions:
  1. Root$^n(\alpha)$: set of the up to n most probable roots starting with $\alpha$
  2. Suffix$^n(\alpha)$: set of the up to n most probable suffixes starting with $\alpha$

- While composing a lemma, the system works in this way:
  ```
  Read(c); \alpha:=<>;
  while c\in A do
    \alpha:= \alpha \cdot c;
    Show(Root^n(\alpha));
    Read(c);
    if Number(c) and 1\leq c \leq Card(Root^n(\alpha)) then
      Complete_lemma(c, \alpha, Root^n(\alpha));
    end if;
  end while;
  Make_updates;
  ```

- While composing a suffix, the system works in this way:
  ```
  Read(c); \alpha:=<>;
  while c\in A do
    \alpha:= \alpha \cdot c;
    Show(Suffix^n(\alpha));
    Read(c);
    if Number(c) and 1\leq c \leq Card(Suffix^n(\alpha)) then
      Complete_suffix(c, \alpha, Suffix^n(\alpha));
    end if;
  end while;
  Make_updates;
  ```
Table 6. Mode of operation of morpheme-based prediction using grammars with acceptance of entire words

- User is modelled as generator of a finite character string
- Objective: to guess the word (root + suffix) the user is going to write
- We include these definitions:
  1. Root\(^n\)(\(\alpha\)): set of the up to \(n\) most probable roots starting with \(\alpha\)
  2. Suffix\(^n\)(\(\alpha\)): set of the up to \(n\) most probable suffixes starting with \(\alpha\)
- Between two space characters, the mode of operation of the system is this:

```plaintext
Read(c); \(\alpha\) :=<>;
while \(c \in A\) do
  \(\alpha\) := \(\alpha\) \cdot \(c\);
  Show(Root\(^n\)(\(\alpha\)));  \(\text{Read}(c)\);
  if NUMBER(\(c\)) and \(1 \leq c \leq \text{Card}(\text{Root}^n(\alpha))\) then
    \(\text{Complete}_\text{word}(\(c\), \(\alpha\), \text{Root}^n(\alpha))\);
  else
    if Number(\(c\)) and \(1 \leq c \leq \text{Card}(\text{Root}^n(\alpha))\) then
      \(\text{Complete}_\text{lemma}(\(c\), \(\alpha\), \text{Root}^n(\alpha))\);
      \(\text{Make}_\text{updates}\);
      \(\text{Read}(c)\); \(\alpha\) :=<>;
      while \(c \in A\) do
        \(\alpha\) := \(\alpha\) \cdot \(c\);
        Show(Suffix\(^n\)(\(\alpha\)));  \(\text{Read}(c)\);
        if NUMBER(\(c\)) and \(1 \leq c \leq \text{Card}(\text{Suffix}^n(\alpha))\) then
          \(\text{Complete}_\text{suffix}(\(c\), \(\alpha\), \text{Suffix}^n(\alpha))\);
        end if;
      end while;
    else
      if Is_equal(\(c\), \text{ROOTEND}) then
        \(\text{Make}_\text{updates}\);
        \(\text{Read}(c)\); \(\alpha\) :=<>;
        while \(c \in A\) do
          \(\alpha\) := \(\alpha\) \cdot \(c\);
          Show(Suffix\(^n\)(\(\alpha\)));  \(\text{Read}(c)\);
          if Number(\(c\)) and \(1 \leq c \leq \text{Card}(\text{Suffix}^n(\alpha))\) then
            \(\text{Complete}_\text{suffix}(\(c\), \(\alpha\), \text{Suffix}^n(\alpha))\);
          end if;
        end while;
      end if;
    end if;
  end if;
end while;
\(\text{Make}_\text{updates}\);
```
Table 7. Example of a Basque sentence without delimiters of morphemes (above) and with them (below), with "^" being the delimiter used.

| Atal hau aipatutako zerbitzuak mantendu eta antolatzeaz arduratuko da, baita erabiltzaile interesatuei beren informazioa unibertsitateko intranet sarean eta interneten eskueran izateko aholkatzeaz ere. |
|---|---|
| Atal^ hau^ aipatutako zerbitzu^ak mantendu^ eta^ antolatze^ ardura^ tuko da^ , baita^ erabiltzaile^ interesatu^ ei beraiek^ en^ informazio^ a^ unibertsitate^ ko^ intranet^ sare^ an^ eta^ internet^ en^ eskuela^ n^ izan^ teko^ aholkatze^ az^ ere^ . |
Figure 1

Keystroke savings (%)

Number of proposals

- Frequencies
- Syntactic+tables
- Syntactic-grammar
Figure 2

![Bar chart showing hit ratio (%) for different number of proposals and syntactic categories]
Figure 3

Keystroke savings (%)

- Words + frequencies
- Morphemes + frequencies
- Morphemes + tables
- Morphemes + grammar
- Total acceptance + grammar

Number of proposals
Figure 4

Number of proposals

Hit ratio (%)

- Words+frequencies
- Morphemes+frequencies
- Morphemes+tables
- Morphemes+grammar
- Total acceptance+grammar