Measurements of the effect of linear interpolation values and reduced bigram model size for text prediction

Marit Ånestad
Lunds Tekniska Högskola
man039@post.uit.no

Michael Geier
Lunds Tekniska Högskola
michael.geier@student.tugraz.at

Abstract

In this paper we present a method for word prediction especially for mobile phones. Based on an existing prediction engine we describe the extensions we implemented to improve its prediction quality and to reduce the size of the data model. We present the results of the benchmarks we ran on the original version of the prediction engine and the results of our improved version. We show out the amount of improvement of the prediction quality by using a more sophisticated algorithm, and the effect on the prediction quality with a model of reduced size.

1 Introduction

Text prediction systems are useful to reduce the amount of keystrokes needed to write texts on computers with alphanumeric keyboards. Especially on mobile devices like mobile phones where writing text is often complicated by the reduced number of keys, text prediction is important to speed up the writing of texts. This paper presents improvements and evaluation of a text prediction system called HMS and our improvements to the text prediction engine. In the evaluation of the test results of the benchmarks we ran on HMS we figure out optimal values for the text prediction algorithm. In our work we focused on the Swedish language, which does not differ much from English or other languages which use the Latin alphabet.

In Section 2, we give an overview of text prediction for mobile devices in general. There we also introduce important terms for talking about text prediction systems and we explain how *bigram models* and *unigram models* can be used for text prediction.

After that, in Section 3, we take a look on an existing text prediction engine called HMS. By *text prediction engine* we mean a software which suggests words the user probably wants to write next based on the letters and/or words written before. In this section we not only describe how the software our work is based on this work, but we also explain the changes we implemented in HMS to improve the quality of the word prediction.

In Section 4 we describe the tests we did to benchmark HMS and we also present the test results to figure out the differences between the original version of HMS and our improved version. Additionally we show how the size of the bigram model influences the prediction quality. When mixing both the bigram model and the unigram model for getting better prediction results we show how assigning different weights to the models influences the prediction results.

Finally, in Section 7 we analyze the test results to point out optimal values for the implemented text prediction algorithms.

2 Text Prediction on Mobile Phones

In this section we describe how text prediction works in general by focusing on the problems of the application on mobile phones. The approach for text prediction we describe in our paper is a frequency based approach (Ganslandt and Jörwall).

The goal of text prediction systems is to reduce the keystrokes needed to write a word or a whole

text. A measure for the relation keystrokes - total characters is *keystrokes per character* or *KSPC*. In other words, the goal of text prediction systems is to minimize KSPC.

$$KSPC = \frac{\sum k}{\sum c}$$

In the equation above k stands for the keystrokes needed to write the desired text including the keystrokes to select the desired words in word suggestion lists. If the user wants to write "home", for example, and if, after typing "home" the suggestion list contains the words "good", "home" and "gone" (in this order), k is 4 (for the three letters "hello") plus 1 (for selecting the second word in the suggestion list. c is the number of total characters of the desired text, which would be "home" in our example. This word has four letters, which leads to a KSPC-value of $5/4 = 1.25$.

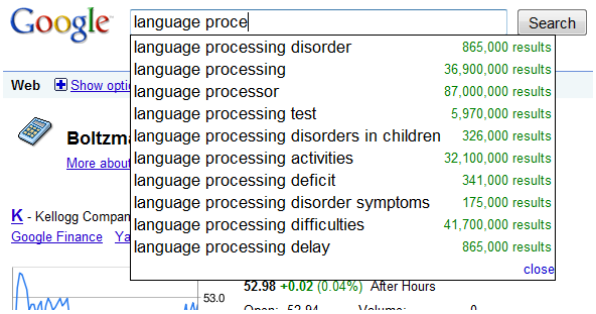

Figure 1: Suggestion list on the Google search page

On most mobile phones, when the user starts writing a text message, a list of suggestions pops up where the user can select the desired word. As there are usually many words in this list, the order of the suggested words is important. The words the user most probably wants to write should be on top of the suggestion list. A measure for how often the desired words are in the beginning of the suggestion list is called *disambiguation accuracy* or *DA*. *DAn* is a measure for how probable it is to have the desired word within the first *n* list entries. DA5, for example, denotes the percentage of having the desired words within the first five list entries when writing a text. This means that the goal of a prediction engine should be to maximize the disambiguation accuracy.

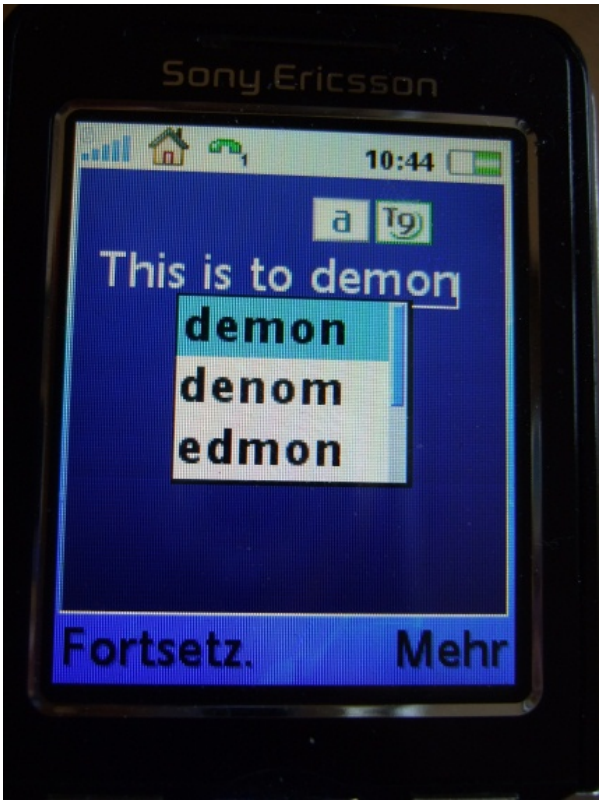

Figure 2: Suggestion list on a mobile phone

As one can see in the screenshots (Figure 1 and 2) there are two different kinds of *text prediction*: Text prediction where the number of letters of the predicted words is equal to the number of letters already typed (Figure 2), and text prediction where the predicted words or phrases have more letters than the text which was already typed (Figure 1). On mobile phones usually the first approach is used. In this approach *KSPC* is always greater than one.

DA1, DA5 and KSPC are values we compare in our test results in Section 4.3.

To create the suggestion lists a dictionary is used. This includes at least a unigram database (or *unigram model*), which is basically a list of single words. An additional bigram database can improve the suggestions. Bigrams are word pairs. By using bigram models the prediction of the current word can be improved by considering the previous word. If a bigram is not in the database the prediction system should fall back to the unigram model.

To create the unigram and the bigram models, a large representative corpus (usually a text file of

plain text) is read from which the number of occurrences of the uni- and bigrams are derived. This has to be done only one time - after this *learning process* the models are trained (which means that the databases are built up).

2.1 Problems

One major problem of text prediction engines is that they can require a large amount of memory to create best results. Especially when using a bigram model the need of memory is high. Usually mobile devices cannot provide such a large amount of memory. This is why the *bigram model*, or in other words the size of the bigram database, should be reduced. In Section 4.3 we will see how a reduced model will influence the prediction quality.

The limited keyboard on most mobile phones is constructed so that one key corresponds to more than one letter. We will examine if a bigram model will work in this context.

It is important to use and evaluate the data in a smart way to get best results. We already mentioned the fallback strategy where first the bigram model is used to create a suggestion list and then, if no bigrams can be found, the unigram model is used. In cases where we can find a bigram, different ways of combining the two models will yield different results, as we will se in Section 4.3.

3 HMS

HMS stands for the three inventors Hasselgren, Montnemery and Svensson (Hasselgren et al.2003). HMS is written in Java and includes the text prediction engine itself and a mobile phone simulator with a typical keyboard where more than one letter corresponds to one key. HMS is written especially for devices with keyboards where more than one letter is mapped to one key. This is done by using a data structure called *Trie*.

There is a graphical user interface to demonstrate the behavior of HMS (see Figure 3), but the main part of HMS is certainly the prediction engine.

The following subsections describe the implementation of HMS. Section 3.1 first describes the state before our modifications, Section 3.2 describes the changes we implemented to improve the prediction engine.

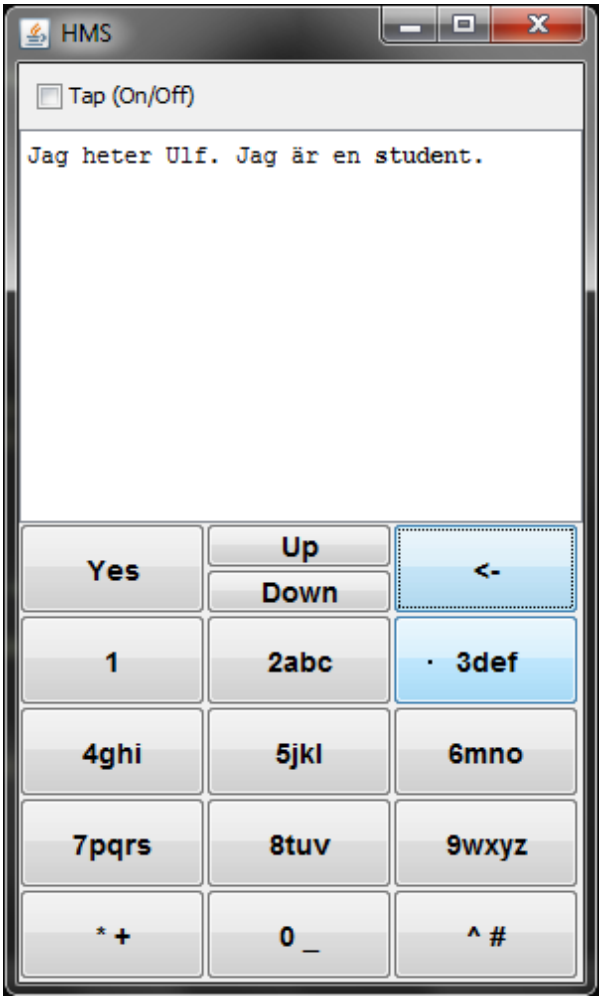

Figure 3: Screenshot of HMS

3.1 Implementation

The text prediction engine of HMS uses a unigram model and a bigram model. The unigram model is represented as Trie data structure (see Section 2.1). Figure 4 illustrates the basic structure of a Trie.

A Trie is a tree of leaves where every leaf represents a key on the keyboard. Every group of siblings (= leaves) consists of one sibling per key. Every leaf contains a list of words which can be created from the key combination when going down the tree starting at the root, and a value for the frequency of the word.

The original HMS had to modes: unigram and bigram mode. The unigram mode used only the unigram model. Each unigram or bigram in the models has a value depending on how often it occur in

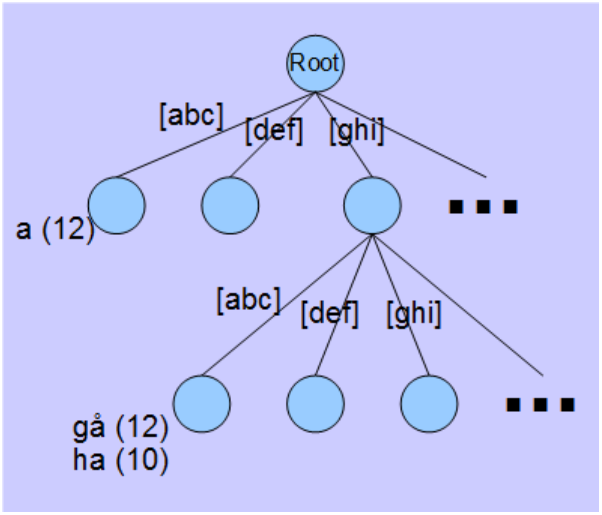

Figure 4: Illustration of the Trie data structure

the training set. The bigram mode gave suggestions both from the bigram and the unigram model using the these values. It ranks all suggestions from the bigram model higher than suggestions from the unigram model. The first mode we call unigram mode, the second we call no interpolation mode

3.2 Changes

We extended the HMS with two more modes: linear interpolation and optimal linear interpolation. The linear interpolation mode uses the unigram and bigram model, as the no interpolation mode, but gives a certain weight to each model. An interpolation of 10 percent multiplies the values of unigram suggestions with 0.1 and value of the bigram suggestions with 0.9. Then the lists are combined and sorted on these values with the highest value at the top of the list. If a word occur in both the unigram and the bigram list, the two values are combined.

The optimal interpolation mode is a mode to measure the efficiency of the linear interpolation. Instead of having a interpolation weight, this mode has an oracle. The oracle always know whether the unigram model or the bigram model will give the best prediction. Given this information, it measures the best case performance of the the linear interpolation method.

4 Tests

We started by writing a framework for automatic testing of the performance of HMS. Tests run on the original HMS implementation gave us a baseline by which we could measure our improvements. Then we tested varying values for bigram cut-off and linear interpolation weights, in order to find optimal values for both parameters.

4.1 Test Setup

The framework must be able to run tests corresponding to the modes we have implemented. The different modes consists of relatively small changes to the test routine, the basis of the tests are the same. Every test goes through the test set, word for word. Each word is translated to a sequence of key strokes on a 9 key keyboard found on most cell phones. This is then the input to the prediction engine. The result is evaluated in regard to the word in the test set.

The *unigram mode* mode uses only the unigram model. The *no optimization* mode uses the combination of unigrams and bigrams of the original HMS with different values for bigram cutoff. The *bigram* mode uses linear interpolation with a given linear interpolation weight for different values for bigram cutoff. The *linear optimization* mode evaluate the prediction with different weight given to the unigram and bigram model. The bigram cut-off is the same for all these tests, but the influence of unigram model vary from 0 - 100 %. The *optimal interpolation* mode produces the best results possible for linear interpolation for one bigram cut-off value.

To accommodate for different types of testing, the testing framework work as a command line tool taking a number of parameters. It is possible to specify the dictionary and training file, the test file, the test type, bigram cut-off value if testing linear interpolation values, linear interpolation value if testing bigram cutoff value and so on.

4.2 Test Data

To be able to test the performance of the system, we needed a dictionary and written text in some language. We chose to use Swedish for our testing. We had a dictionary file of 118 414 Swedish words words. For the written text we used Wikipedia entries.

The Wikipedia dump of all pages in a language gives a large corpus. It is not annotated, but that was no need of our approach. It displays current usage of a language, and has many authors, thus reducing the bias of individual authors. The genre of the text, though, is quite different than what can be expected from sms-messages. The articles describes and explains different phenomena in a scientific and neutral manner, whereas sms messages might contain more everyday words, dialogue, emotional language and questions. To overcome this, a corpus of sms's or similar communication must be gathered.

We downloaded the newest dump of the whole Swedish Wikipedia as of December 6th 2009 (Wikipedia2009). This dump was treated according to the instructions and programs of (Mudge2009) to remove formatting and non text elements. Then the dump was split into sentences. Any sentence containing non Latin 1 characters was discarded. This was necessary, due to a high number of words written in other scripts which the HMS system did not handle well.

From the refined dump the training, test and development set were extracted. This happened by taking every 160 sentence for each set, with 6 sentences between each set.This reduces the influence of any specific article on the sets, and also reduces the similarity between the different sets. The training set were used in all tests. The test set were used to in tests where the bigram cutoff varies, whereas the development set is used in tests where the linear interpolation varies.

After removing the formatting from the Wikipedia dump it contained about 45 million words. Our test, training and development set contained each about 280 000 words. We chose to use this size because larger data sets would increased the time to run tests and because of the limited memory and computational power on our machines.

4.3 Test Results

First we ran tests on the prediction quality of the original HMS. Figures 5 and 6 shows that the KSPC value increases and the DA1 value decreases with increased bigram cutoff value. The largest model gives the best prediction as expected. What we did not expect was that the unigram mode outperforms the bigram mode, if only by less than 0,2%.

4.3.1 HMS Bigrams vs. Unigrams

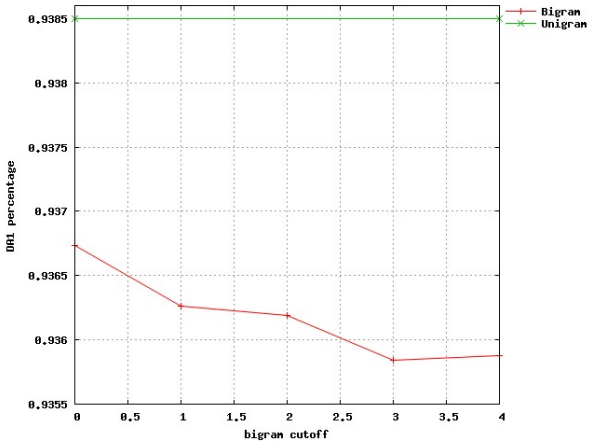

Figure 5: HMS without modifications - DA1 in relation to bigram cut-off value

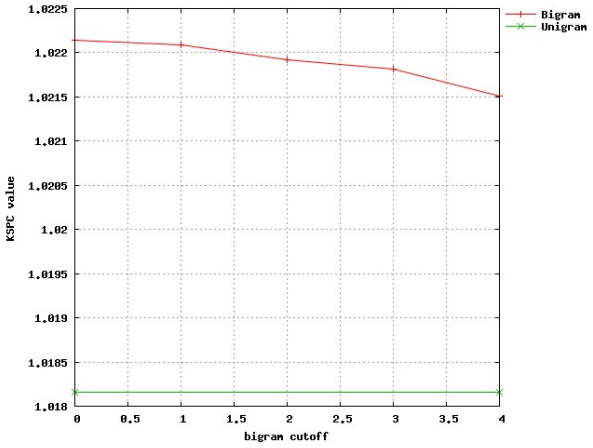

Figure 6: HMS without modifications - KSPC in relation to bigram cut-off value

4.3.2 Linear Interpolation

Then we implemented linear interpolation. All these test has a bigram cut-off value of 5, except the HMS value.

The results for DA1 is found in Figure 7. The uppermost line show the result of optimal interpolation mode: 0.945%. This is the best we can do with combinations of the bigram and unigram model. As a baseline we use the original HMS performance with a cutoff of 4, which gives a DA1 of 0.936%. The

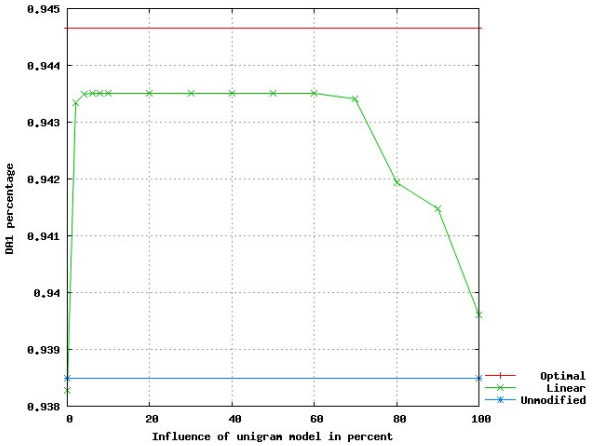

Figure 7: DA1 in relation to linear interpolation value

unigram mode gives results slightly better than the original HMS: 0.938%. The results for the linear interpolation varies with different weight given to the unigram and bigram models. All these values lies between the optimal and the baseline result, except the instance where the bigram model is the sole model used. The prediction engine gives the best results where the unigram model get between 4% and 60% weight. This DA1 result of 0.944% is only 0.1% below the optimal result, and an improvement from the unigram mode by 0.5%.

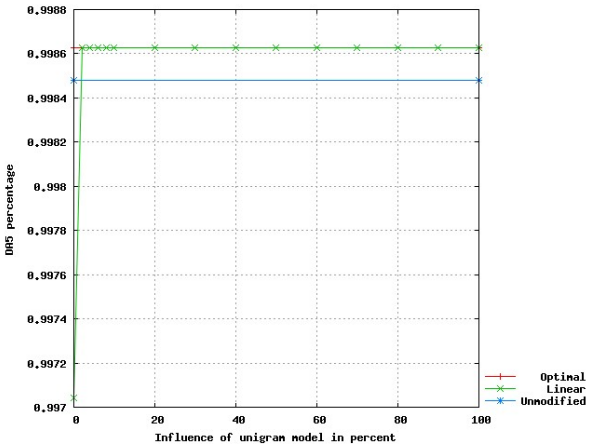

Figure 8: DA5 in relation to linear interpolation value

In Figure 8 we see that the linear interpolation value has less influence on the DA5 metric. Any combination except only the bigram model gives the modified HMS the same results as the results from our optimal interpolation mode. The improvement from the unmodified HMS is small, only a little more than 0.01%.

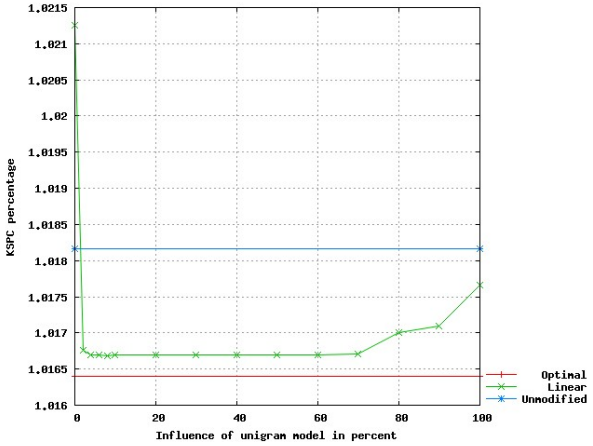

Figure 9: KSPC in relation to linear interpolation value

Figure 9 show the same tendencies for the KSPC metric as 7 did for the DA1 metric. The original HMS has a KSPC value of 1.0215, the unigram model a KSPC of 1.0182 and the the optimal interpolation a value of 1.0164. We get the best results with between 4% and 70% weight to the unigram model. This gives KSPC values of 1.0167, 0.0048 below the original HMS, 0.0015 below the unigram model and only 0.0003 above the optimal value.

4.3.3 Bigrams: Best cut-off value

When we had found good interpolation values, we started to investigate the effect on the prediction when we reduced the size of the bigram model. The full model is about 186 000 bigrams, about 83 000 kb in size. We also have about 150 000 unigrams, about 22 000 kb in size. Figure 10 show the size of the model in number of bigrams with different bigram cut-off values. Just removing the bigrams that occur only once reduces the model by more than 85%. A cut-off of 5 gives a reduction of the model of almost 98%, to the size of 1660 kb.

All these tests where done with an linear interpolation unigram weight of 2%. The figure 11 and 12 show the effect on the DA1 and KSPC metrics. As expected, the DA1 results decreases with a smaller model and the KSPC value increases. This decrease

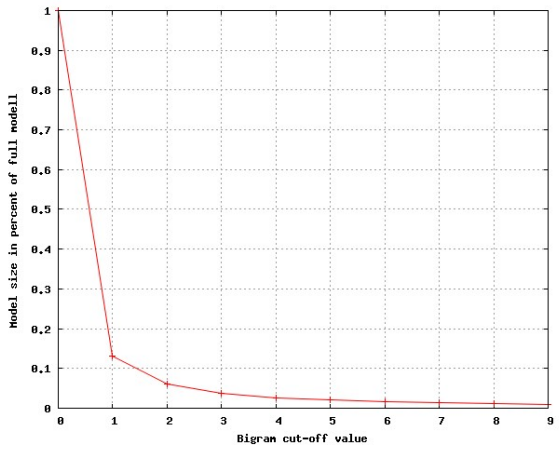

Figure 10: Model size in relation to bigram cut-off value

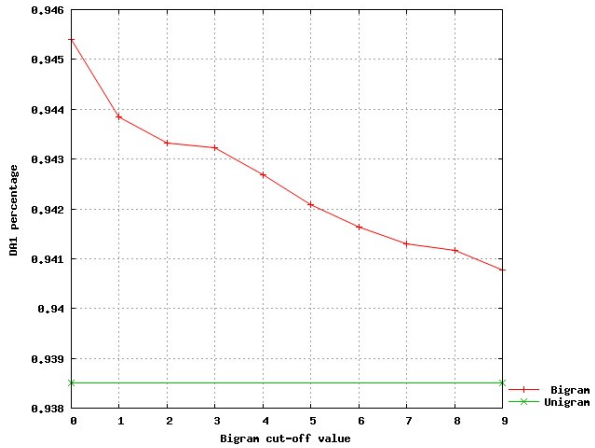

Figure 11: DA1 in relation to bigram cutoff value

and increase are roughly linear in regard to the cut-off value, while the size of the model is not. A large reduction of the size of the model gives less of a reduction in the prediction performance.

The DA5 value shows almost no change: the difference between the best and worst value is only 0.007%.

Figure 13 shows the prediction results with different bigram cut-off values relative to the results of the full bigram model. This graph shows that the results for a bigram cuf-off value of 5 is only 0.4%worse than the result from the full bigram model. The KSPC value is only 0.1% higher compared to the result from the full model.

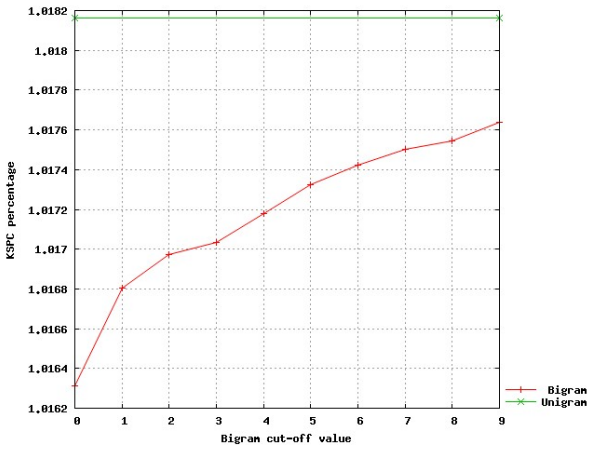

Figure 12: KSPC in relation to bigram cut-off value

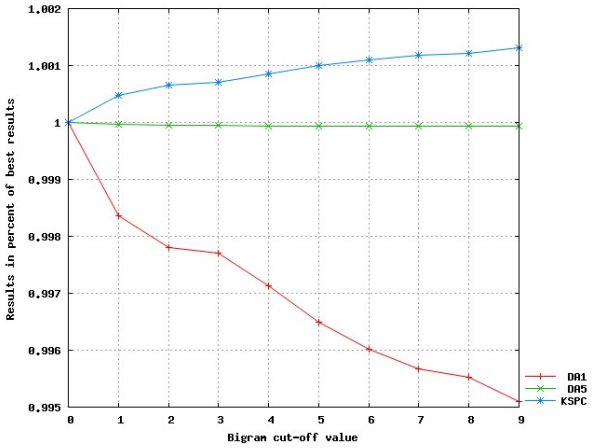

Figure 13: Relative results in relation to bigram cut-off value

5 Discussion and future work

Our work can be improved in several ways. First, we have not limited our unigram model enough. Currently it consists of all words in the dictionary and the training file, while ideally it should be pruned like we do with the bigram model. Secondly, our training and test files have only about 300 000 words. Larger training files will probably give a better model, which can be reduced as we have shown. Third, our corpus is not ideal for applications for text messages. Our corpus consist of factual texts, where sms-messages probably will consist of personal and everyday language.

Our experiments with bigram cut-off shows that it is possible to reduce the size of the bigram model to

a feasible size and still get improvements. It is possible to get better prediction by spending a little more memory on the device to hold the bigram model. But using linear interpolation will use more computation on the devices. In also requires a training effort on a suitable corpus in each language.

6 Related work

There is much work done on input methods for mobile phones. We have used a bigram model and linear interpolation. (Hasselgren et al.2003) reviews a number of different text entry methods, focusing on letter based methods, including LetterWise, which predicts letters based on the previous letters. The paper also explains the development of HMS. (Höjer2008) uses n-grams to translate Hindi written in English to Hindi written in its own script Devanagari. (Selander and Svensson2009) describes a text prediction engine for Indic scripts with large alphabeth on a input device with few buttens. It employs part of speech tags in addition to a bigram model. (Ganslandt and Jörwall) employs syntax and semantic information together with a bigram model for text prediction on a standard mobile phone keyboards. (Gong et al.2008) uses the same approach as Ganslandt and JÃ¶rwall, but with more investigation on the effect on even more limited keyboards.

7 Conclusion

We have showed that a text prediction engine that combines a bigram and unigram model with linear interpolation way gives better results than a engine that employs only a bigram model or a unigram model. In our tests, we got the best results when the unigram model got between 4% and 60% influence. In addition, we showed that a bigram model can be reduced by more than 98% and still gain an improvement compared to a unigram or bigram only model.

References

Sebastian Ganslandt and Jakob Jörwall. Context-aware predictive text entry for swedish using semantics and syntax.

Jun Gong, Peter Tarasewich, and Scott MacKenzie. 2008. Improved word list ordering for text entry on ambigous keypads. *Proceedings of the Fifth Nordic Conference on Human-Computer Interaction - NordiCHI 2008*, pages 152–161.

Jon Hasselgren, Erik Montnemery, and Markus Svensson. 2003. Hms: A predictive text entry method using bigrams.

Magnus Höjer. 2008. Phonetic text input for indic scripts.

Raphael Mudge. 2009. Generating a plain text corpus from wikipedia. Blog article at After the Deadline, December.

Mitch Selander and Erik Svensson. 2009. Predictive text input engine for indic scripts.

Wikipedia. 2009. Wikipedia database dump, December.