Predicting Sentences using N-Gram Language Models

Steffen Bickel, Peter Haider, and Tobias Scheffer
Humboldt-Universität zu Berlin
Department of Computer Science
Unter den Linden 6, 10099 Berlin, Germany
{bickel, haider, scheffer}@informatik.hu-berlin.de

Abstract

We explore the benefit that users in several application areas can experience from a "tab-complete" editing assistance function. We develop an evaluation metric and adapt N-gram language models to the problem of predicting the subsequent words, given an initial text fragment. Using an instance-based method as baseline, we empirically study the predictability of call-center emails, personal emails, weather reports, and cooking recipes.

1 Introduction

Prediction of user behavior is a basis for the construction of assistance systems; it has therefore been investigated in diverse application areas. Previous studies have shed light on the predictability of the next unix command that a user will enter (Motoda and Yoshida, 1997; Davison and Hirsch, 1998), the next keystrokes on a small input device such as a PDA (Darragh and Witten, 1992), and of the translation that a human translator will choose for a given foreign sentence (Nepveu et al., 2004).

We address the problem of predicting the subsequent words, given an initial fragment of text. This problem is motivated by the perspective of assistance systems for repetitive tasks such as answering emails in call centers or letters in an administrative environment. Both instance-based learning and N-gram models can conjecture completions of sentences. The use of N-gram models requires the application of the Viterbi principle to this particular decoding problem.

Quantifying the benefit of editing assistance to a user is challenging because it depends not only on an observed distribution over documents, but also on the reading and writing speed, personal preference, and training status of the user. We develop an evaluation metric and protocol that is practical, intuitive, and independent of the user-specific tradeoff between keystroke savings and time lost due to distractions. We experiment on corpora of service-center emails, personal emails of an Enron executive, weather reports, and cooking recipes.

The rest of this paper is organized as follows. We review related work in Section 2. In Section 3, we discuss the problem setting and derive appropriate performance metrics. We develop the N-gram-based completion method in Section 4. In Section 5, we discuss empirical results. Section 6 concludes.

2 Related Work

Shannon (1951) analyzed the predictability of sequences of letters. He found that written English has a high degree of redundancy. Based on this finding, it is natural to ask whether users can be supported in the process of writing text by systems that predict the intended next keystrokes, words, or sentences. Darragh and Witten (1992) have developed an *interactive keyboard* that uses the sequence of past keystrokes to predict the most likely succeeding keystrokes. Clearly, in an unconstrained application context, keystrokes can only be predicted with limited accuracy. In the specific context of entering URLs, completion predictions are commonly pro-

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 193–200, Vancouver, October 2005.

vided by web browsers (Debevc et al., 1997).

Motoda and Yoshida (1997) and Davison and Hirsch (1998) developed a Unix shell which predicts the command stubs that a user is most likely to enter, given the current history of entered commands. Korvemaker and Greiner (2000) have developed this idea into a system which predicts entire command lines. The Unix command prediction problem has also been addressed by Jacobs and Blockeel (2001) who infer macros from frequent command sequences and predict the next command using variable memory Markov models (Jacobs and Blockeel, 2003).

In the context of *natural language*, several typing assistance tools for apraxic (Garay-Vitoria and Abascal, 2004; Zagler and Beck, 2002) and dyslexic (Magnuson and Hunnicutt, 2002) persons have been developed. These tools provide the user with a list of possible word completions to select from. For these users, scanning and selecting from lists of proposed words is usually more efficient than typing. By contrast, scanning and selecting from many displayed options can slow down skilled writers (Langlais et al., 2002; Magnuson and Hunnicutt, 2002).

Assistance tools have furthermore been developed for translators. Computer aided translation systems combine a translation and a language model in order to provide a (human) translator with a list of suggestions (Langlais et al., 2000; Langlais et al., 2004; Nepveu et al., 2004). Foster et al. (2002) introduce a model that adapts to a user's typing speed in order to achieve a better trade-off between distractions and keystroke savings. Grabski and Scheffer (2004) have previously developed an indexing method that efficiently retrieves the sentence from a collection that is most similar to a given initial fragment.

3 Problem Setting and Evaluation

Given an initial text fragment, a predictor that solves the sentence completion problem has to conjecture *as much of the sentence that the user currently intends to write*, as is possible with high confidence—preferably, but not necessarily, the entire remainder.

The perceived benefit of an assistance system is highly subjective, because it depends on the expenditure of time for scanning and deciding on suggestions, and on the time saved due to helpful assistance. The user-specific benefit is influenced by quantitative factors that we can measure. We construct a system of two conflicting performance indicators: our definition of *precision* quantifies the inverse risk of unnecessary distractions, our definition of *recall* quantifies the rate of keystroke savings.

For a given sentence fragment, a completion method may – but need not – cast a completion conjecture. Whether the method suggests a completion, and how many words are suggested, will typically be controlled by a confidence threshold. We consider the entire conjecture to be falsely positive if at least one word is wrong. This harsh view reflects previous results which indicate that selecting, and then editing, a suggested sentence often takes longer than writing that sentence from scratch (Langlais et al., 2000). In a conjecture that is entirely accepted by the user, the entire string is a true positive. A conjecture may contain only a part of the remaining sentence and therefore the *recall*, which refers to the length of the missing part of the current sentence, may be smaller than 1.

For a given test collection, precision and recall are defined in Equations 1 and 2. *Recall* equals the fraction of saved keystrokes (disregarding the interface-dependent single keystroke that is most likely required to accept a suggestion); *precision* is the ratio of characters that the users have to scan for each character they accept. Varying the confidence threshold of a sentence completion method results in a *precision recall curve* that characterizes the system-specific trade-off between *keystroke savings* and *unnecessary distractions*.

$$Precision = \frac{\sum_{\text{accepted completions}} \text{string length}}{\sum_{\text{suggested completions}} \text{string length}} \quad (1)$$

$$Recall = \frac{\sum_{\text{accepted completions}} \text{string length}}{\sum_{\text{all queries}} \text{length of missing part}} \quad (2)$$

4 Algorithms for Sentence Completion

In this section, we derive our solution to the sentence completion problem based on linear interpolation of N-gram models. We derive a k best Viterbi decoding algorithm with a confidence-based stopping criterion which conjectures the words that most likely succeed an initial fragment. Additionally, we

briefly discuss an instance-based method that provides an alternative approach and baseline for our experiments.

In order to solve the sentence completion problem with an N-gram model, we need to find the most likely word sequence $w_{t+1}, \ldots, w_{t+T}$ given a word N-gram model and an initial sequence $w_1, \ldots, w_t$ (Equation 3). Equation 4 factorizes the joint probability of the missing words; the N-th order Markov assumption that underlies the N-gram model simplifies this expression in Equation 5.

$$\operatorname*{argmax}_{w_{t+1},\ldots,w_{t+T}} P(w_{t+1}, \ldots, w_{t+T}|w_1, \ldots, w_t) \quad (3)$$

$$= \operatorname*{argmax}_{w_{t+1},\ldots,w_{t+T}} \prod_{j=1}^{T} P(w_{t+j}|w_1, \ldots, w_{t+j-1}) \quad (4)$$

$$= \operatorname{argmax} \prod_{j=1}^{T} P(w_{t+j}|w_{t+j-N+1}, \ldots, w_{t+j-1}) \quad (5)$$

The individual factors of Equation 5 are provided by the model. The Markov order N has to balance sufficient context information and sparsity of the training data. A standard solution is to use a weighted linear mixture of N-gram models, $1 \leq n \leq N$, (Brown et al., 1992). We use an EM algorithm to select mixing weights that maximize the generation probability of a tuning set of sentences that have not been used for training.

We are left with the following questions: (a) how can we decode the most likely completion *efficiently*; and (b) how many words should we predict?

4.1 Efficient Prediction

We have to address the problem of finding the most likely completion, $\operatorname{argmax}_{w_{t+1},\ldots,w_{t+T}} P(w_{t+1}, \ldots, w_{t+T}|w_1, \ldots, w_t)$ *efficiently*, even though the size of the *search space* grows exponentially in the number of predicted words.

We will now identify the recursive structure in Equation 3; this will lead us to a Viterbi algorithm that retrieves the most likely word sequence. We first define an auxiliary variable $\delta_{t,s}(w'_1, \ldots, w'_N|w_{t-N+2}, \ldots, w_t)$ in Equation 6; it quantifies the greatest possible probability over all arbitrary word sequences $w_{t+1}, \ldots, w_{t+s}$, followed by the word sequence $w_{t+s+1} = w'_1, \ldots, w_{t+s+N} = w'_N$, conditioned on the initial word sequence $w_{t-N+2}, \ldots, w_t$.

In Equation 7, we factorize the last transition and utilize the N-th order Markov assumption. In Equation 8, we split the maximization and introduce a new random variable w'_0 for w_{t+s}. We can now refer to the definition of δ and see the recursion in Equation 9: $\delta_{t,s}$ depends only on $\delta_{t,s-1}$ and the N-gram model probability $P(w'_N|w'_1, \ldots, w'_{N-1})$.

$$\delta_{t,s}(w'_1, \ldots, w'_N|w_{t-N+2}, \ldots, w_t) \quad (6)$$

$$= \max_{w_{t+1},\ldots,w_{t+s}} P(w_{t+1}, \ldots, w_{t+s}, w_{t+s+1} = w'_1, \ldots, w_{t+s+N} = w'_N|w_{t-N+2}, \ldots, w_t)$$

$$= \max_{w_{t+1},\ldots,w_{t+s}} P(w'_N|w'_1, \ldots, w'_{N-1}) \quad (7)$$
$$P(w_{t+1}, \ldots, w_{t+s}, w_{t+s+1} = w'_1, \ldots, w_{t+s+N-1} = w'_{N-1}|w_{t-N+2}, \ldots, w_t)$$

$$= \max_{w'_0} \max_{w_{t+1},\ldots,w_{t+s-1}} P(w'_N|w'_1, \ldots, w'_{N-1}) \quad (8)$$
$$P(w_{t+1}, \ldots, w_{t+s-1}, w_{t+s} = w'_0, \ldots, w_{t+s+N-1} = w'_{N-1}|w_{t-N+2}, \ldots, w_t)$$

$$= \max_{w'_0} P(w'_N|w'_1, \ldots, w'_{N-1}) \quad (9)$$
$$\delta_{t,s-1}(w'_0, \ldots, w'_{N-1}|w_{t+N-2}, \ldots, w_t)$$

Exploiting the N-th order Markov assumption, we can now express our target probability (Equation 3) in terms of δ in Equation 10.

$$\max_{w_{t+1},\ldots,w_{t+T}} P(w_{t+1}, \ldots, w_{t+T}|w_{t-N+2}, \ldots, w_t) \quad (10)$$
$$= \max_{w'_1,\ldots,w'_N} \delta_{t,T-N}(w'_1, \ldots, w'_N|w_{t-N+2}, \ldots, w_t)$$

The last N words in the most likely sequence are simply the $\operatorname{argmax}_{w'_1,\ldots,w'_N} \delta_{t,T-N}(w'_1, \ldots, w'_N|w_{t-N+2}, \ldots, w_t)$. In order to collect the preceding most likely words, we define an auxiliary variable Ψ in Equation 11 that can be determined in Equation 12. We have now found a Viterbi algorithm that is linear in T, the completion length.

$$\Psi_{t,s}(w'_1, \ldots, w'_N|w_{t-N+2}, \ldots, w_t) \quad (11)$$

$$= \operatorname*{argmax}_{w_{t+s}} \max_{w_{t+1},\ldots,w_{t+s-1}} P(w_{t+1}, ..., w_{t+s}, w_{t+s+1} = w'_1, ..., w_{t+s+N} = w'_N|w_{t-N+2}, ..., w_t)$$

$$= \operatorname*{argmax}_{w'_0} \frac{\delta_{t,s-1}(w'_0, \ldots, w'_{N-1}|w_{t-N+2}, \ldots, w_t)}{P(w'_N|w'_1, \ldots, w'_{N-1})} \quad (12)$$

The Viterbi algorithm starts with the most recently entered word w_t and moves iteratively into the future. When the N-th token in the highest scored δ is a period, then we can stop as our goal is only to predict (parts of) the current sentence. However, since

there is no guarantee that a period will eventually become the most likely token, we use an absolute confidence threshold as additional criterion: when the highest δ score is below a threshold θ, we stop the Viterbi search and fix T.

In each step, Viterbi stores and updates |vocabulary size|^N many δ values—unfeasibly many except for very small N. Therefore, in Table 1 we develop a Viterbi beam search algorithm which is linear in T and in the beam width. Beam search cannot be guaranteed to always find the most likely word sequence: When the globally most likely sequence $w^*_{t+1}, \ldots, w^*_{t+T}$ has an initial subsequence $w^*_{t+1}, \ldots, w^*_{t+s}$ which is not among the k most likely sequences of length s, then that optimal sequence is not found.

Table 1: Sentence completion with Viterbi beam search algorithm.

Input: N-gram language model, initial sentence fragment $w_1, \ldots, w_t$, beam width k, confidence threshold θ.

1. Viterbi initialization:
   Let $\delta_{t,-N}(w_{t-N+1}, \ldots, w_t | w_{t-N+1}, \ldots, w_t) = 1$;
   let $s = -N + 1$;
   $beam(s-1) = \{\delta_{t,-N}(w_{t-N+1}, \ldots, w_t | w_{t-N+1}, \ldots, w_t)\}$.
2. **Do** Viterbi recursion **until** break:
   (a) **For** all $\delta_{t,s-1}(w'_0, \ldots, w'_{N-1} | \ldots)$ in $beam(s-1)$, **for** all w_N in vocabulary, store $\delta_{t,s}(w'_1, \ldots, w'_N | \ldots)$ (Equation 9) in $beam(s)$ and calculate $\Psi_{t,s}(w'_1, \ldots, w'_N | \ldots)$ (Equation 12).
   (b) **If** $\text{argmax}_{w_N} \max_{w'_1, \ldots, w'_{N-1}} \delta_{t,s}(w'_1, \ldots, w'_N | \ldots) =$ *period* **then** break.
   (c) **If** $\max \delta_{t,s}(w'_1, \ldots, w'_N | w_{t-N+1}, \ldots, w_t) < \theta$ **then** decrement s; break.
   (d) Prune all but the best k elements in $beam(s)$.
   (e) Increment s.
3. **Let** $T = s + N$. Collect words by path backtracking:
   $(w^*_{t+T-N+1}, \ldots, w^*_{t+T}) = \text{argmax}\, \delta_{t,T-N}(w'_1, \ldots, w'_N | ...)$.
   For $s = T - N \ldots 1$:
   $w^*_{t+s} = \Psi_{t,s}(w^*_{t+s+1}, \ldots, w^*_{t+s+N} | w_{t-N+1}, \ldots, w_t)$.

Return $w^*_{t+1}, \ldots, w^*_{t+T}$.

4.2 Instance-based Sentence Completion

An alternative approach to sentence completion based on N-gram models is to retrieve, from the training collection, the sentence that starts most similarly, and use its remainder as a completion hypothesis. The cosine similarity of the TFIDF representation of the initial fragment to be completed, and an equally long fragment of each sentence in the training collection gives both a selection criterion for the nearest neighbor and a confidence measure that can be compared against a threshold in order to achieve a desired precision recall balance.

A straightforward implementation of this nearest neighbor approach becomes infeasible when the training collection is large because too many training sentences have to be processed. Grabski and Scheffer (2004) have developed an indexing structure that retrieves the most similar (using cosine similarity) sentence fragment in sub-linear time. We use their implementation of the instance-based method in our experimentation.

5 Empirical Studies

we investigate the following questions. (a) How does sentence completion with N-gram models compare to the instance-based method, both in terms of precision/recall and computing time? (b) How well can N-gram models complete sentences from collections with diverse properties?

Table 2 gives an overview of the four document collections that we use for experimentation. The first collection has been provided by a large online store and contains emails sent by the service center in reply to customer requests (Grabski and Scheffer, 2004). The second collection is an excerpt of the recently disclosed email correspondence of Enron's management staff (Klimt and Yang, 2004). We use 3189 personal emails sent by Enron executive Jeff Dasovich; he is the individual who sent the largest number of messages within the recording period.

The third collection contains textual daily weather reports for five years from a weather report provider on the Internet. Each report comprises about 20 sentences. The last collection contains about 4000 cooking recipes; this corpus serves as an example of a set of thematically related documents that might be found on a personal computer.

We reserve 1000 sentences of each data set for testing. As described in Section 4, we split the remaining sentences in training (75%) and tuning

Table 2: Evaluation data collections.

Name	Language	#Sentences	Entropy
service center	German	7094	1.41
Enron emails	English	16363	7.17
weather reports	German	30053	4.67
cooking recipes	German	76377	4.14

(25%) sets. We mix N-gram models up to an order of five and estimate the interpolation weights (Section 4). The resulting weights are displayed in Figure 1. In Table 2, we also display the entropy of the collections based on the interpolated 5-gram model. This corresponds to the average number of bits that are needed to code each word given the preceding four words. This is a measure of the intrinsic redundancy of the collection and thus of the predictability.

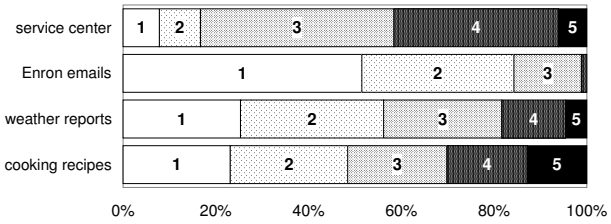

Figure 1: N-gram interpolation weights.

Our evaluation protocol is as follows. The beam width parameter k is set to 20. We randomly draw 1000 sentences and, within each sentence, a position at which we split it into initial fragment and remainder to be predicted. A human evaluator is presented both, the actual sentence from the collection and the initial fragment plus current completion conjecture. For each initial fragment, we first cast the most likely single word prediction and ask the human evaluator to judge whether they would accept this prediction (without any changes), given that they intend to write the actual sentence. We increase the length of the prediction string by one additional word and recur, until we reach a period or exceed the prediction length of 20 words.

For each judged prediction length, we record the confidence measure that would lead to that prediction. With this information we can determine the results for all possible threshold values of θ. To save evaluation time, we consider all predictions that are identical to the actual sentence as correct and skip those predictions in the manual evaluation.

We will now study how the N-gram method compares to the instance-based method. Figure 2 compares the precision recall curves of the two methods. Note that the maximum possible recall is typically much smaller than 1: recall is a measure of the keystroke savings, a value of 1 indicates that the user saves *all* keystrokes. Even for a confidence threshold of 0, a recall of 1 is usually not achievable.

Some of the precision recall curves have a concave shape. Decreasing the threshold value increases the number of predicted words, but it also increases the risk of at least one word being wrong. In this case, the entire sentence counts as an incorrect prediction, causing a decrease in both, precision and recall. Therefore – unlike in the standard information retrieval setting – recall does not increase monotonically when the threshold is reduced.

For three out of four data collections, the instance-based learning method achieves the highest maximum recall (whenever this method casts a conjecture, the entire remainder of the sentence is predicted—at a low precision), but for nearly all recall levels the N-gram model achieves a much higher precision. For practical applications, a high precision is needed in order to avoid distracting, wrong predictions. Varying the threshold, the N-gram model can be tuned to a wide range of different precision recall trade-offs (in three cases, precision can even reach 1), whereas the confidence threshold of the instance-based method has little influence on precision and recall.

We determine the standard error of the precision for the point of maximum F1-measure. For all data collections and both methods the standard error is below 0.016. Correct and incorrect prediction examples are provided in Table 3 for the service center data set, translated from German into English. The confidence threshold is adjusted to the value of maximum F1-measure. In two of these cases, the prediction nicely stops at fairly specific terms.

How do precision and recall depend on the string length of the initial fragment and the string length of the completion cast by the systems? Figure 3 shows the relationship between the length of the initial fragment and precision and recall. The performance of the instance-based method depends crucially on a long initial fragment. By contrast, when

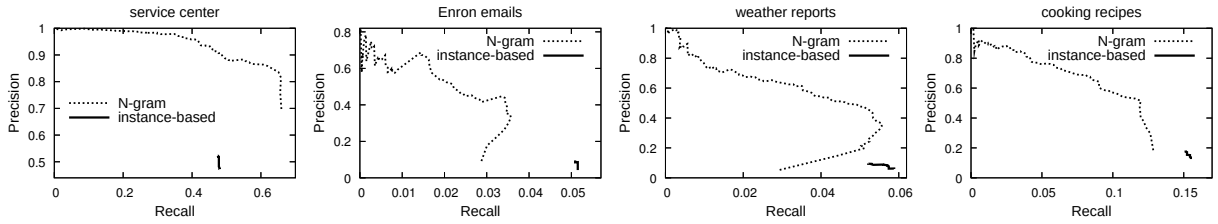

Figure 2: Precision recall curves for N-gram and instance-based methods of sentence completion.

Table 3: Prediction examples for service center data.

Initial fragment (bold face) and intended, missing part	Prediction
Please complete your address.	your address.
Kindly excuse the incomplete shipment.	excuse the
Our supplier notified us that the pants are undeliverable.	notified us that the
The mentioned order is not in our system.	not in our system.
We recommend that you write down your login name and password.	that you write down your login name and password.
The value will be accounted for in your invoice.	be accounted for in your invoice.
Please excuse the delay.	delay.
Please excuse our mistake.	the delay.
If this is not the case give us a short notice.	us your address and customer id.

the fragment length exceeds four with the N-gram model, then this length and the accuracy are nearly independent; the model considers no more than the last four words in the fragment.

Figure 4 details the relation between string length of the prediction and precision/recall. We see that we can reach a constantly high precision over the entire range of prediction lengths for the service center data with the N-gram model. For the other collections, the maximum prediction length is 3 or 5 words in comparison to much longer predictions cast by the nearest neighbor method. But in these cases, longer predictions result in lower precision.

How do instance-based learning and N-gram completion compare in terms of computation time? The Viterbi beam search decoder is linear in the prediction length. The index-based retrieval algorithm is constant in the prediction length (except for the final step of *displaying* the string which is linear but can be neglected). This is reflected in Figure 5 (left) which also shows that the absolute decoding time of both methods is on the order of few milliseconds on a PC. Figure 5 (right) shows how prediction time grows with the training set size.

We experiment on four text collections with diverse properties. The N-gram model performs remarkably on the service center email collection. Users can save 60% of their keystrokes with 85% of all suggestions being accepted by the users, or save 40% keystrokes at a precision of over 95%. For cooking recipes, users can save 8% keystrokes at 60% precision or 5% at 80% precision. For weather reports, keystroke savings are 2% at 70% correct suggestions or 0.8% at 80%. Finally, Jeff Dasovich of Enron can enjoy only a marginal benefit: below 1% of keystrokes are saved at 60% entirely acceptable suggestions, or 0.2% at 80% precision.

How do these performance results correlate with properties of the model and text collections? In Figure 1, we see that the mixture weights of the higher order N-gram models are greatest for the service center mails, smaller for the recipes, even smaller for the weather reports and smallest for Enron. With 50% of the mixture weights allocated to the 1-gram model, for the Enron collection the N-gram completion method can often only guess words with high prior probability. From Table 2, we can furthermore see that the entropy of the text collection is inversely proportional to the model's ability to solve the sentence completion problem. With an entropy

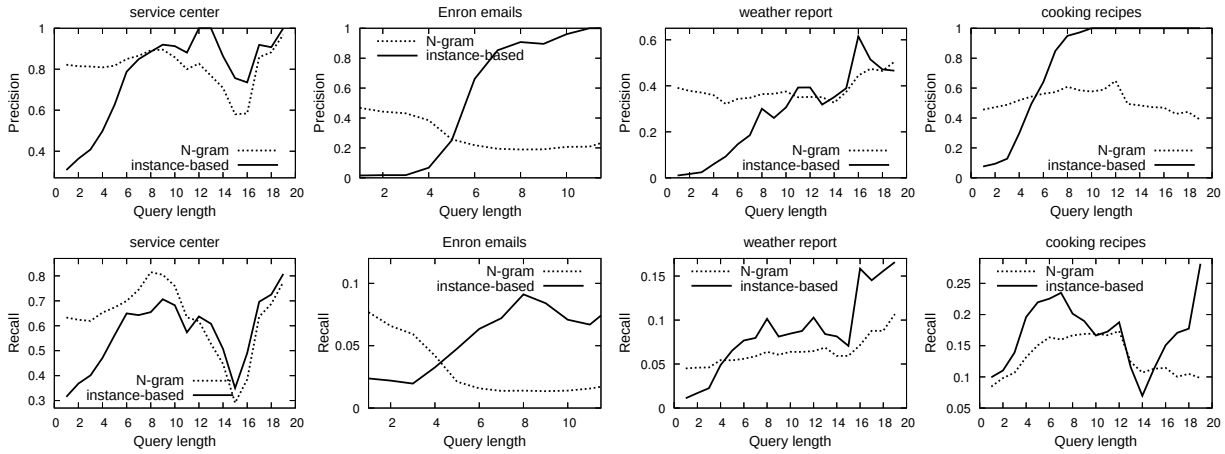

Figure 3: Precision and recall dependent on string length of initial fragment (words).

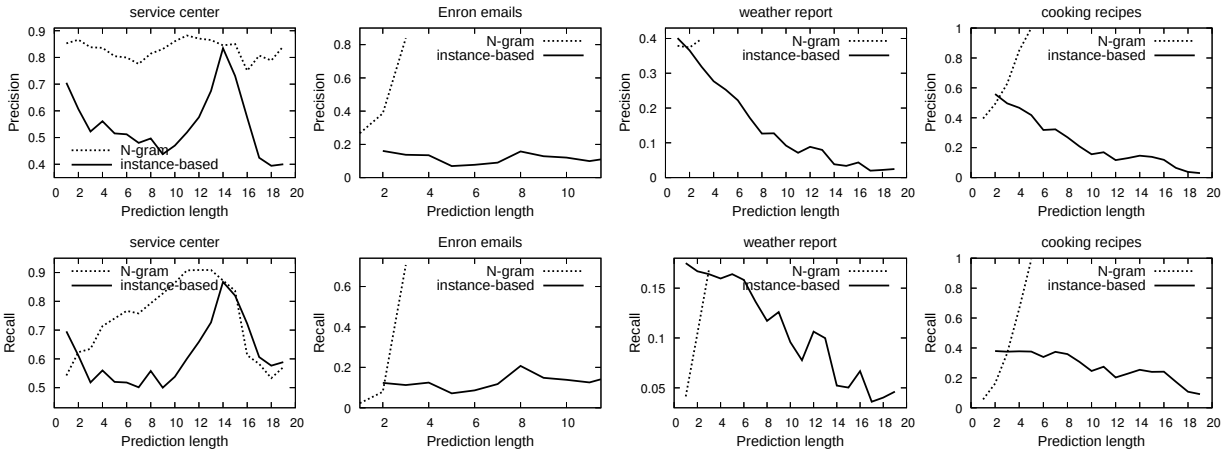

Figure 4: Precision and recall dependent on prediction string length (words).

of only 1.41, service center emails are excellently predictable; by contrast, Jeff Dasovich's personal emails have an entropy of 7.17 and are almost as unpredictable as Enron's share price.

6 Conclusion

We discussed the problem of predicting how a user will complete a sentence. We find precision (the number of suggested characters that the user has to read for every character that is accepted) and recall (the rate of keystroke savings) to be appropriate performance metrics. We developed a sentence completion method based on N-gram language models. We derived a k best Viterbi beam search decoder. Our experiments lead to the following conclusions:

(a) The N-gram based completion method has a better precision recall profile than index-based retrieval of the most similar sentence. It can be tuned to a wide range of trade-offs, a high precision can be obtained. The execution time of the Viterbi beam search decoder is in the order of few milliseconds.

(b) Whether sentence completion is helpful strongly depends on the diversity of the document collection as, for instance, measured by the entropy. For service center emails, a keystroke saving of 60% can be achieved at 85% acceptable suggestions; by contrast, only a marginal keystroke saving of 0.2% can be achieved for Jeff Dasovich's personal emails at 80% acceptable suggestions. A modest but significant benefit can be observed for thematically related documents: weather reports and cooking recipes.

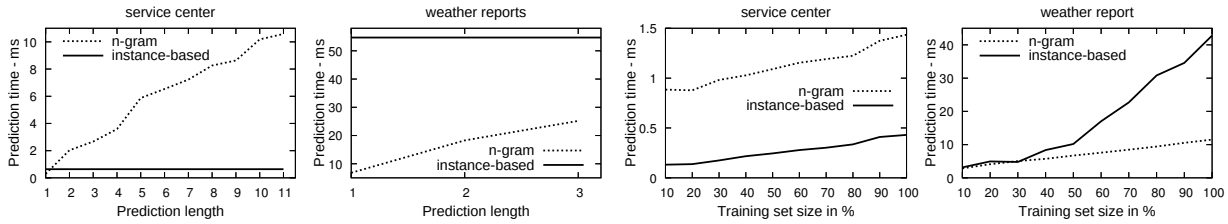

Figure 5: Prediction time dependent on prediction length in words (left) and prediction time dependent on training set size (right) for *service center* and *weather report* collections.

Acknowledgment

This work has been supported by the German Science Foundation DFG under grant SCHE540/10.

References

P. Brown, S. Della Pietra, V. Della Pietra, J. Lai, and R. Mercer. 1992. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(2):31–40.

J. Darragh and I. Witten. 1992. *The Reactive Keyboard*. Cambridge University Press.

B. Davison and H. Hirsch. 1998. Predicting sequences of user actions. In *Proceedings of the AAAI/ICML Workshop on Predicting the Future: AI Approaches to Time Series Analysis*.

M. Debevc, B. Meyer, and R. Svecko. 1997. An adaptive short list for documents on the world wide web. In *Proceedings of the International Conference on Intelligent User Interfaces*.

G. Foster, P. Langlais, and G. Lapalme. 2002. User-friendly text prediction for translators. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

N. Garay-Vitoria and J. Abascal. 2004. A comparison of prediction techniques to enhance the communication of people with disabilities. In *Proceedings of the 8th ERCIM Workshop User Interfaces For All*.

K. Grabski and T. Scheffer. 2004. Sentence completion. In *Proceedings of the ACM SIGIR Conference on Information Retrieval*.

N. Jacobs and H. Blockeel. 2001. The learning shell: automated macro induction. In *Proceedings of the International Conference on User Modelling*.

N. Jacobs and H. Blockeel. 2003. Sequence prediction with mixed order Markov chains. In *Proceedings of the Belgian/Dutch Conference on Artificial Intelligence*.

B. Klimt and Y. Yang. 2004. The Enron corpus: A new dataset for email classification research. In *Proceedings of the European Conference on Machine Learning*.

B. Korvemaker and R. Greiner. 2000. Predicting Unix command lines: adjusting to user patterns. In *Proceedings of the National Conference on Artificial Intelligence*.

P. Langlais, G. Foster, and G. Lapalme. 2000. Unit completion for a computer-aided translation typing system. *Machine Translation*, 15:267–294.

P. Langlais, M. Loranger, and G. Lapalme. 2002. Translators at work with transtype: Resource and evaluation. In *Proceedings of the International Conference on Language Resources and Evaluation*.

P. Langlais, G. Lapalme, and M. Loranger. 2004. Transtype: Development-evaluation cycles to boost translator's productivity. *Machine Translation (Special Issue on Embedded Machine Translation Systems*, 17(17):77–98.

T. Magnuson and S. Hunnicutt. 2002. Measuring the effectiveness of word prediction: The advantage of long-term use. Technical Report TMH-QPSR Volume 43, Speech, Music and Hearing, KTH, Stockholm, Sweden.

H. Motoda and K. Yoshida. 1997. Machine learning techniques to make computers easier to use. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*.

L. Nepveu, G. Lapalme, P. Langlais, and G. Foster. 2004. Adaptive language and translation models for interactive machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

C. Shannon. 1951. Prediction and entropy of printed english. In *Bell Systems Technical Journal, 30, 50-64.*

W. Zagler and C. Beck. 2002. FASTY - faster typing for disabled persons. In *Proceedings of the European Conference on Medical and Biological Engineering*.