Learning to Complete Sentences*

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 consider the problem of predicting how a user will continue a given initial text fragment. Intuitively, our goal is to develop a "tab-complete" function for natural language, based on a model that is learned from text data. We consider two learning mechanisms that generate predictive models from collections of application-specific document collections: we develop an N-gram based completion method and discuss the application of instance-based learning. After developing evaluation metrics for this task, we empirically compare the model-based to the instance-based method and assess the predictability of call-center emails, personal emails, and weather reports.

1 Introduction

This paper addresses the problem of predicting the succeeding words of an initial fragment of natural language text. This problem setting is motivated by applications that include repetitive tasks such as writing emails in call centers or letters in an administrative environment; many resulting documents are to some degree governed by specific underlying patterns that can be learned. The benefit of an assistance system to the user depends on both the number of helpful suggestions and the number of unnecessary distractions that they experience. Performance metrics for other text learning problems do not match the idiosyncrasies of this problem; we therefore have to discuss an appropriate evaluation scheme.

Generative N-gram language models provide a natural approach to the construction of sentence completion systems; in addition, instance-based learning can easily be applied to this problem. In order to gain insights on the benefit of sentence completion methods in various application areas, we conduct experiments using diverse document collections: a collection of call-center emails with many similar emails, weather reports, cooking recipes, and, on the other extreme, the disclosed Enron corpus of personal email communication of Enron's management staff.

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 appropriate performance metrics. We develop the N-gram-based completion method in Section 4. In Section 5, we discuss empirical results. Section 6 concludes.

* *Proceedings of the European Conference on Machine Learning.* 2005.

2 Related Work

Shannon [13] 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 it is possible to support users in the process of writing text by learning and predicting the intended next keystrokes, words, sentences, or even paragraphs. Darragh and Witten [1] 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 provided by web browsers [3].

Motoda and Yoshida [12] and Davison and Hirsh [2] 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 [9] 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 who predict the next command using variable memory Markov models [7].

In the context of *natural language*, several typing assistance tools for apraxic [5, 14] and dyslexic [11] persons have been developed. These tools provide the user with a list of possible word completions to select from. For these particular 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 [10, 11]. Assistance tools have furthermore been developed for translators. Computer aided translation systems combine a translation model and a language model in order to provide a (human) translator with a list of suggestions [4]. Grabski and Scheffer [6] 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

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 corresponding learning problem is to find such a predictor, given a corpus that is governed by the same distribution of sentences.

What is an appropriate performance measure for this problem? The perceived benefit of an assistance system is highly subjective, but it is influenced by quantitative factors that we can measure. We define a system of two conflicting performance indicators: *precision* (the inverse risk of unnecessary distractions) and *recall* (the keystroke savings). Keystroke savings obtained by accepting suggestions and distraction caused by (rejected) suggestions contribute to the overall benefit of a system. However, the exact trade-off between the increased typing speed due to saved keystrokes and the time lost because of distractions is highly user-specific; any measurement of the actual time savings is a projection of these conflicting goals for a particular group of users.

For a given sentence fragment, a completion method may, but need not, cast a completion conjecture. Whether – and how many – words are suggested will typically be controlled by a confidence threshold. Given an individual sentence fragment, 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 can take longer than writing that sentence from scratch [10]. In a conjecture that is entirely accepted by the user, the entire string is a true positive. Note that 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* is 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*.

$$Precision = \frac{\sum_{\text{accepted completions}} \text{string length}}{\sum_{\text{suggested completions}} \text{string length}} \tag{1}$$

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

4 Algorithms for Sentence Completion

We derive our solution to the sentence completion problem based on a linear interpolation of N-gram models, and 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 $w_1, \ldots, w_t$; that is, we need to decode $\text{argmax}_{w_{t+1},\ldots,w_{t+T}} P(w_{t+1}, \ldots, w_{t+T} | w_1, \ldots, w_t)$. The N-th order Markov assumption constrains each w_t to be dependent on at most w_{t-N+1} through w_{t-1}. The individual $P(w_t | w_{t-N+1}, \ldots, w_{t-1})$ are the parameters of the model.

Learning Linearly Interpolated N-Gram Models
Given a fixed Markov order N and a document collection, an N-gram model is learned by estimating the probability of all combinations of N words.

One solution to overcome sparsity problems of higher order N-grams is to use a weighted linear mixture of N-gram models, $1 \leq n \leq N$, Equation 3.

$$P(w_N | w_1, \ldots, w_{N-1}) = \lambda_1 P(w_N) + \sum_{n=2}^{N} \lambda_n P(w_N | w_{N-n+1}, \ldots, w_{N-1}) \tag{3}$$

We learn Models for all n, $1 \leq n \leq N$ on a training fraction, adjust the parameters λ_n such that the likelihood of a hold-out fraction is maximized, and retrain the N-gram models with these fixed λ_n on the entire data collection.

Efficient Decoding
We have to find the most likely completion, $\text{argmax}_{w_{t+1},\ldots,w_{t+T}} P(w_{t+1},\ldots,w_{t+T}|w_1,\ldots,w_t)$ *efficiently*, even though the size of the *search space* is $|\text{vocabulary size}|^T$. The auxiliary variable $\delta_{t,s}(w'_1,\ldots,w'_N|w_{t-N+2},\ldots,w_t)$ in Equation 4 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 $w_{t-N+2},\ldots,w_t$. Equation 5 utilizes the N-th order Markov assumption, in Equation 6, we 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 7.

$$\delta_{t,s}(w'_1,\ldots,w'_N|w_{t-N+2},\ldots,w_t) \tag{4}$$
$$= \max_{w_{t+1},\ldots,w_{t+s}} P(w_{t+1},\ldots,w_{t+s},w_{t+s+1}=w'_1,\ldots,w_{t+s+N}=w'_N|w_{t-N+2},\ldots,w_t)$$
$$= \max_{w_{t+1},\ldots,w_{t+s}} P(w'_N|w'_1,\ldots,w'_{N-1}) \tag{5}$$
$$P(w_{t+1},\ldots,w_{t+s},w_{t+s+1}=w'_1,\ldots,w_{t+s+N-1}=w'_{N-1}|w_{t-N+2},\ldots,w_t)$$
$$= \max_{w'_0} \max_{w_{t+1},\ldots,w_{t+s-1}} P(w'_N|w'_1,\ldots,w'_{N-1}) \tag{6}$$
$$P(w_{t+1},\ldots,w_{t+s-1},w_{t+s}=w'_0,\ldots,w_{t+s+N-1}=w'_{N-1}|w_{t-N+2},\ldots,w_t)$$
$$= \max_{w'_0} P(w'_N|w'_1,\ldots,w'_{N-1})\delta_{t,s-1}(w'_0,\ldots,w'_{N-1}|w_{t+N-2},\ldots,w_t) \tag{7}$$

Exploiting the N-th order Markov assumption, we can now express our target probability in terms of δ in Equation 8.

$$\max_{w_{t+1},\ldots,w_{t+T}} P(w_{t+1},\ldots,w_{t+T}|w_{t-N+2},\ldots,w_t) \tag{8}$$
$$= \max_{w'_1,\ldots,w'_N} \delta_{t,T-N}(w'_1,\ldots,w'_N|w_{t-N+2},\ldots,w_t)$$

The last N words in the most likely sequence are $\text{argmax}_{w'_1,\ldots,w'_N} \delta_{t,T-N}(w'_1,\ldots,w'_N|w_{t-N+2},\ldots,w_t)$. Variable Ψ, defined in Equation 9 collects the preceding most likely words; it can be determined in Equation 10. We have now found a Viterbi algorithm that is linear in T.

$$\Psi_{t,s}(w'_1,\ldots,w'_N|w_{t-N+2},\ldots,w_t) \tag{9}$$
$$= \underset{w_{t+s}}{\text{argmax}} \max_{w_{t+1},\ldots,w_{t+s-1}} P(w_{t+1},\ldots,w_{t+s},w_{t+s+1}=w'_1,\ldots,w_{t+s+N}=w'_N|w_{t-N+2},\ldots,w_t)$$
$$= \text{argmax}_{w'_0}\ \delta_{t,s-1}(w'_0,\ldots,w'_{N-1}|w_{t-N+2},\ldots,w_t)\ P(w'_N|w'_1,\ldots,w'_{N-1}) \tag{10}$$

The Viterbi algorithm starts with the most recently entered word w_t and moves iteratively into the future. The process terminates when the N-th token in the highest scored δ is a period, or when the highest δ score is below a threshold θ. In each step, Viterbi has to store and update $|\text{vocabulary size}|^N$ 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. 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 7) in $beam(s)$ and calculate $\Psi_{t,s}(w'_1, \ldots, w'_N | \ldots)$ (Equation 10).
   (b) **If** $\text{argmax}_{w_N} \max_{w'_1, \ldots, w'_{N-1}} \delta_{t,s}(w'_1, \ldots, w'_N | \ldots) = \textit{period}$ **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 $w^*_{t+1}, \ldots, w^*_{t+T}$.

Instance-based Sentence Completion

An alternative to 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. Grabski and Scheffer [6] have developed an indexing structure that retrieves the most similar (in terms of cosine similarity) sentence fragment in sub-linear time.

5 Empirical Studies

We investigate (a) how N-gram models compare to the instance-based method in terms of precision/recall; (b) how well N-gram models complete sentences from collections with diverse properties. We use four document collections. The first contains emails sent by the customer center of a large online store [6] (7094 sentences). The second contains 3189 personal emails sent by Enron executive Jeff Dasovich, extracted from the Enron email corpus [8]. The third collection contains textual daily weather reports of five years. The last collection contains about 4000 cooking recipes.

We reserve 1000 sentences of each data set for testing and split the remaining sentences in training (75%) and tuning (25%) set for λ_n. We mix N-gram models up to an order of five. The beam width k is set to 20. We randomly draw 1000

sentences and a position at which we split it into initial fragment and remainder to be predicted. We loop over the number of words to predict, decode the most likely completion of that length, and provide a human evaluator with both, the true sentence as well as the initial fragment plus the current completion conjecture. The judges decide whether they would accept this prediction without any changes, given that they intend to write the actual sentence. For each judged prediction length, we record the threshold that would lead to that prediction; thus, we determine precision and recall for all thresholds.

Figure 1 compares the precision recall curves of the N-gram and instance-based methods. Note that the maximum possible recall is typically much smaller than 1. Recall measures the rate of keystroke savings; a value of 1 indicates that the user saves *all* keystrokes. Some of the precision recall curves have a concave

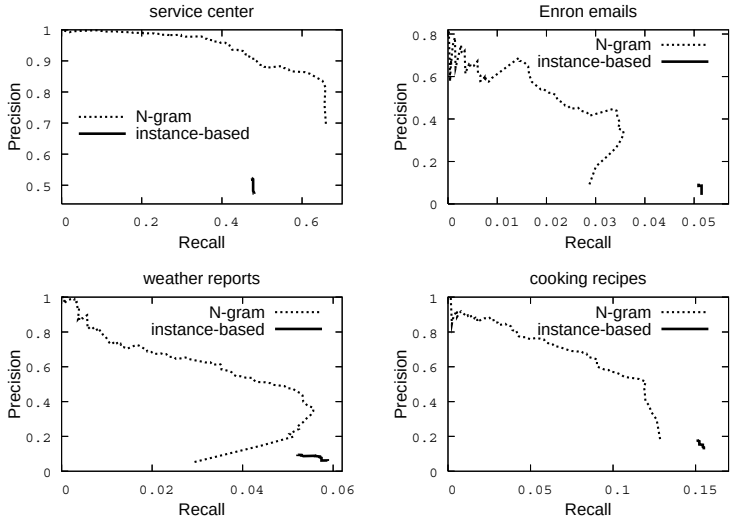

Fig. 1. Precision recall curves for N-gram and instance-based methods.

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 data collections, the instance-based learning method achieves the highest maximum recall (each conjecture predicts the entire remainder of the sentence—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 avoids distracting, wrong predictions. The N-gram model can be tuned to a wide range of different precision recall trade-offs (precision can often reach 1), whereas the threshold has little influence on precision and recall of the instance-based method. The standard error of the measurements is below 0.016.

How do precision and recall depend on the string length of the initial fragment? Figure 2 details this relationship. The performance of the instance-based method depends crucially on a long initial fragment. The N-gram model works best when the initial fragment is at least four $(N-1)$, but the model does not benefit from additional tokens.

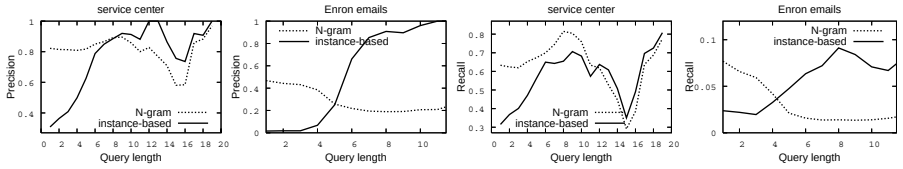

Fig. 2. Precision and recall dependent on string length of initial fragment (words).

The four text collections have diverse properties. The N-gram model performs remarkably on the service center emails. Users can save 60% of their keystrokes with 85% of all suggestions being entirely 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, possible 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% precision, or 0.2% at 80% precision.

We observe that these performance results correlate with the entropy of the data sets, and with the N-gram mixture weights. The service center data has an entropy of only 1.41 (*i.e.,* 1.41 bits are needed, on average, to encode the next token, given the four preceding tokens); they are excellently predictable. A mixing weight of 40% is assigned to the 3-gram, 30% to the 4-gram and 10% to the 5-gram model. The entropy of the cooking recipes and the weather reports are 4.14 and 4.67, respectively. About 25% probability mass are assigned to the each of the unigram, bigram, and trigram model.

For the Enron data, $\lambda_1 = 50\%$ is assigned to the unigram and $\lambda_2 = 30\%$ to the bigram; hence, the model can often just guess stop words that have a high prior probability. Jeff Dasovich's personal emails have an entropy of 7.17 and are therefore almost as unpredictable as Enron's share price. This argues that the entropy of a text collection is an excellent measure that indicates whether, for a given discourse area, a user will benefit from sentence completion.

6 Conclusion

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 for the problem of predicting subsequent words in a given initial fragment. We developed an efficient sentence completion method based on N-gram language models.

Our experiments lead to two main conclusions. (a) The N-gram based completion method has a better precision recall profile than the nearest neighbor method. It can be tuned to a wide range of trade-offs, a high precision can be obtained. (b) Whether sentence completion is helpful strongly depends on the diversity of the document collection. 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. The entropy of the text is a strong indicator of the potential benefit of sentence completion that can easily be measured.

Acknowledgment. This work has been supported by the German Science Foundation DFG under grants SCHE540/10-1 and SCHE540/10-2.

References

1. J. Darragh and I. Witten. *The Reactive Keyboard.* Cambridge University Press, 1992.
2. B. Davison and H. Hirsh. Predicting sequences of user actions. In *AAAI/ICML Workshop on Predicting the Future: AI Approaches to Time Series Analysis*, 1998.
3. M. Debevc, B. Meyer, and R. Svecko. An adaptive short list for documents on the world wide web. In *Proceedings of the International Conference on Intelligent User Interfaces*, 1997.
4. G. Foster. *Text Prediction for Translators.* PhD thesis, University of Montreal, 2002.
5. N. Garay-Vitoria and J. Abascal. A comparison of prediction techniques to enhance the communication of people with disabilities. In *Proceedings of the 8th ERCIM Workshop User Interfaces For All*, 2004.
6. K. Grabski and T. Scheffer. Sentence completion. In *Proceedings of the ACM SIGIR Conference on Information Retrieval*, 2004.
7. N. Jacobs and H. Blockeel. User modelling with sequential data. In *Proceedings of the HCI International*, 2003.
8. B. Klimt and Y. Yang. The Enron corpus: A new dataset for email classification research. In *Proceedings of the European Conference on Machine Learning*, 2004.
9. B. Korvemaker and R. Greiner. Predicting Unix command lines: adjusting to user patterns. In *Proceedings of the National Conference on Artificial Intelligence*, 2000.
10. P. Langlais, M. Loranger, and G. Lapalme. Translators at work with transtype: Resource and evaluation. In *Proceedings of the third international Conference on Language Resources and Evaluation*, 2002.
11. T. Magnuson and S. Hunnicutt. 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, 2002.
12. H. Motoda and K. Yoshida. Machine learning techniques to make computers easier to use. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1997.
13. C. Shannon. Prediction and entropy of printed english. In *Bell Systems Technical Journal, 30, 50-64*, 1951.
14. W. Zagler and C. Beck. FASTY - faster typing for disabled persons. In *Proceedings of the European Conference on Medical and Biological Engineering*, 2002.