IMPROVEMENTS IN STOCHASTIC LANGUAGE MODELING

Ronald Rosenfeld and Xuedong Huang

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

We describe two attempt to improve our stochastic language models. In the first, we identify a systematic overestimation in the traditional backoff model, and use statistical reasoning to correct it. Our modification results in up to 6% reduction in the perplexity of various tasks. Although the improvement is modest, it is achieved with hardly any increase in the complexity of the model. Both analysis and empirical data suggest that the modification is most suitable when training data is sparse.

In the second attempt, we propose a new type of adaptive language model. Existing adaptive models use a dynamic cache, based on the history of the document seen up to that point. But another source of information in the history, within-document word sequence correlations, has not yet been tapped. We describe a model that attempts to capture this information, using a framework where one word sequence triggers another, causing its estimated probability to be raised. We discuss various issues in the design of such a model, and describe our first attempt at building one. Our preliminary results include a perplexity reduction of between 10% and 32%, depending on the test set.

1. INTRODUCTION

Linguistic constraints are an important factor in human comprehension of speech. Their effect on automatic speech recognition is similar, in that they provide both a pruning method and a means of ordering likely candidates. As vocabularies for speech recognition systems increase in size, more accurate modeling of linguistic constraints becomes essential.

Two fundamental issues in language modeling are smoothing and adaptation. Smoothing allows a model to assign reasonable probabilities to events that have never been observed before. Adaptation takes advantage of recently gained knowledge — the text seen so far — to adjust the model's expectations.

In what follows, we discuss two attempts at improving our current stochastic language modeling techniques. In the first, we try to improve smoothing by correcting a deficiency in a successful and well known smoothing method, the backoff model. In the second, we propose a novel kind of adaptation, one that is based on correlation among word sequences occurring in the same document.

2. CORRECTING OVERESTIMATION IN THE BACKOFF MODEL

2.1. The Problem

The backoff n-gram language model[1] estimates the probability of w_n given the immediate past history $w_1^{n-1} = (w_1, \ldots w_{n-1})$. It is defined recursively as:

$$P_n(w_n|w_1^{n-1}) = \begin{cases} (1-d)C(w_1^n) \,/\, C(w_1^{n-1}) & \text{if } C(w_1^n) > 0 \\ \alpha(C(w_1^{n-1})) \cdot P_{n-1}(w_n|w_2^{n-1}) & \text{if } C(w_1^n) = 0 \end{cases} \quad (1)$$

where d, the discount ratio, is a function of $C(w_1^n)$, and the α's are the backoff weights, calculated to satisfy the sum-to-1 probability constraints.

The backoff language model is a compact yet powerful way of modeling the dependence of the current word on its immediate history. An important factor in the backoff model is its behavior on the backed-off cases, namely when a given n-gram w_1^n is found not to have occurred in the training data. In these cases, the model assumes that the probability is proportional to the estimate provided by the n-1–gram, $P_{n-1}(w_n|w_2^{n-1})$.

This last assumption is reasonable most of the time, since no other sources of information are available. But for frequent n-1–grams, there may exist sufficient statistical evidence to suggest that the backed-off probabilities should in fact be much lower. This phenomenon occurs at any value of n, but is easiest to demonstrate for the simple case of $n = 2$, i.e. a bigram. Consider the following fictitious but typical example:

N = 1,000,000
C("ON") = 10,000
C("AT") = 10,000
C("CALL") = 100
C("ON","AT") = 0
C("ON","CALL") = 0

N is the total number of words in the training set, and $C(w_i, w_j)$ is the number of (w_i, w_j) bigrams occurring in that set. The backoff model computes:

P("AT") = $\frac{1}{100}$
P("CALL") = $\frac{1}{10,000}$
P("AT"|"ON") = α("ON") · P("AT") = α("ON")· $\frac{1}{100}$
P("CALL"|"ON") = α("ON")·P("CALL") = α("ON")· $\frac{1}{10000}$

Thus, according to this model, P("AT"|"ON") $\gg$ P("CALL"|"ON"). But this is clearly incorrect. In the case of "CALL", the expected number of ("ON","CALL") bigrams, assuming independence between "ON" and "CALL", is 1, so an actual count of 0 does not give much information, and may be ignored. However, in the case of "AT", the expected chance count of ("ON","AT") is 100, so an actual count of 0 means that the real probability of P("AT"|"ON") is in fact much lower than chance. The backoff model does not capture this information, and thus grossly overestimates P("AT"| "ON").

This deficiency of the backoff model has been pointed out before[2, p.457], but, to the best of our knowledge, has never been corrected. We suspect the reasons are twofold. First, it only occurs during backed-off cases. For a well trained bigram or trigram, this happens in only a small fraction of the time. Second, overestimation degrades perplexity only mildly and indirectly, by affecting a slight underestimation of all the other probabilities.

We therefore did not expect this phenomenon to have a strong impact on perplexity. Nevertheless, we wanted to correct the problem and to measure its effect.

2.2. The Solution: Confidence Interval Capping

Let $C(w_1^n) = 0$. Given a global confidence level Q, to be determined empirically, we calculate a confidence interval in which the true value of $P(w_n|w_1^{n-1})$ should lie, using the constraint:

$$[1 - P(w_n|w_1^{n-1})]^{C(w_1^{n-1})} \geq Q \qquad (2)$$

The confidence interval is therefore $[0 \ldots (1 - Q^{1/C(w_1^{n-1})})]$. We then provide another parameter, P $(0 < P \leq 1)$, and establish a ceiling, or a *cap*, at a point P within the confidence interval:

$$\mathrm{CAP}_{Q,P}(C(w_1^{n-1})) = P \cdot (1 - Q^{1/C(w_1^{n-1})}) \qquad (3)$$

We now require that the estimated $P(w_n|w_1^{n-1})$ satisfy:

$$P(w_n|w_1^{n-1}) \leq \mathrm{CAP}_{Q,P}(C(w_1^{n-1})) \qquad (4)$$

The backoff case of the standard model is therefore modified to:

$$P(w_n|w_1^{n-1}) = \min\left[\alpha(w_1^{n-1}) \cdot P_{n-1}(w_n|w_2^{n-1}),\ \mathrm{CAP}_{Q,P}(C(w_1^{n-1}))\right] \qquad (5)$$

This *capping off* of the estimates requires renormalization. But renormalization would increase the α's, which would in turn cause some backed-off probabilities to exceed the cap. An iterative reestimation of the α's is therefore required. The process was found to converge rapidly in all cases.

Note that, although some computation is required to determine the new weights, once the model has been computed, it is no more complicated neither significantly more time consuming than the original one.

2.3. Results

The bigram perplexity reduction for various tasks is shown in table 1. BC-48K is the brown corpus with the unabridged

test set	backoff rate	PP reduction
BC-48K	30%	6.3%
BC-5K	15%	2.5%
ATIS	5%	1.7%
WSJ-5K	2%	0.8%

Table 1: Perplexity reduction by Confidence Interval Capping

vocabulary of 48,455 words. BC-5K is the same corpus, restricted to the most frequent 5,000 words. ATIS is the class-based bigram developed at CMU for the ATIS task. WSJ is the official CSR 5c.vp task.

Although the reduction is modest, as expected, it should be remembered that it is achieved with hardly any increase in the complexity of the model. As can be predicted from the statistical analysis, when the vocabulary is larger, the backoff rate is greater, and the improvement in perplexity can be expected to be greater too.

3. TRIGGER-BASED ADAPTATION

3.1. Motivation and Analysis

Several adaptive language models have been proposed recently [3, 4, 5, 6], which use caching of the partially dictated document, and interpolate a dynamic component based on the cache with the static component. These models have been successful in reducing the perplexity of the text considerably, and [5] also reports a positive effect on the word recognition rate.

All of these models make direct use of the words in the history of the document. They take advantage of the fact that words, and combinations of words, once occurred in a given document, have a higher likelihood of occurring in it again.

But there is another source of information in the history that has not yet been tapped: within-document correlation between

words or word sequences. Consider the sentence:

> "The district attorney's office launched a comprehensive investigation into loans made by several well connected banks."

Based on this sentence alone, a cache-based model will not be able to anticipate any of the constituent words. But a human reader might use "DISTRICT ATTORNEY" and/or "LAUNCHED" to anticipate "INVESTIGATION", and "LOANS" to anticipate "BANKS".

In what follows, we describe a model that attempts to capture this type of information in a systematic way, using correlation between word sequences derived from a large corpus of text. In this model, if a word sequence A is positively and significantly correlated with another word sequence B, then $(A \rightarrow B)$ is considered a "trigger pair", with A being the trigger and B the triggered sequence. When A occurs in the document, it triggers B, causing its probability estimate to be increased.

In order for such a model to be effective, the following issues have to be addressed:

1. *How to filter all possible trigger pairs.* Even if we restrict our attention to pairs where A and B are both single words, the number of such pairs is too large. Let V be the size of the vocabulary. Note that, unlike in a bigram model, where the number of different consecutive word pairs is much less than V^2, the number of word pairs where both words occurred in the same document is a significant fraction of V^2.
2. *How to combine evidence from multiple triggers.* This is a special case of the general problem of combining evidence from several sources. We discuss several heuristics, and a plan for a more disciplined approach.
3. *How to combine the triggering model with the static model.*

We will discuss all 3 problems and our proposed solutions to them. This is ongoing research, and not all of our ideas have been tested yet. A solution to (1) will be discussed in some detail. When combined with simple minded solutions to (2) and (3), it resulted in a perplexity reduction of between 10% and 32%, depending on the test set. We are currently working on implementing and testing some of the other solutions.

3.2. Filtering the Trigger-Pairs

Let "history" denote the part of the text already seen by the system. Let A, B be any two word sequences. Then the events B and B_0 are defined as follows:

B : B occurred in the history.
B_0 : B occurs next in the document.

Let $P(B_0)$ be the (unconditional) probability of B_0, and let $P(B_0|A)$ be the conditional probability assigned to B_0 by the trigger pair $(A \rightarrow B)$. A natural measure of the information provided by A on B_0 is the mutual information between the two:

$$I(A : B_0) = \log \frac{P(B_0|A)}{P(B_0)} \tag{6}$$

Note that, although mutual information is symmetric with regard to its arguments, it is generally not true that $I(A : B_0) = I(B : A_0)$.

Should mutual information be our figure of merit in selecting the most promising trigger pairs? $I(A : B_0)$ measures the average number of bits we can save by considering A in predicting B_0. But this savings will materialize only if B_0 is true, namely if we indeed encounter the word sequence B next in the document. Our best estimate of this, at the time filtering is carried out, is $P(B_0|A)$. We therefore define *the expected utility* of the trigger pair $(A \rightarrow B)$:

$$U(A \rightarrow B) \stackrel{\text{def}}{=} I(A : B_0) P(B_0|A) \tag{7}$$

and suggest it as a criterion for selecting trigger pairs.

3.3. Multiply-Triggered Sequences

The problem of combining evidence from multiple sources is a general, largely unsolved problem in modeling. The ideal solution is to model explicitly each combination of values of the predictor variables, but this leads to an exponential growth in the number of parameters, which renders the model untrainable. At the other extreme, we can assume linearity and simply sum the contribution from the different sources. This may be a reasonable approximation in some models, but it is clearly inadequate in our case: "LOAN" is not 3 times more likely after 3 occurrences of "BANK" than it is after only 1 occurrence.

Multiple triggers have several important functions:

- Increase the reliability of the prediction in the face of unreliable history. Since we usually rely on the speech recognizer to provide us with the history, each word has a nonnegligible chance of being erroneous.
- Disambiguate multiple-sense words.
  Compare:
  P("LOAN"$_0$|"BANK")
  P("LOAN"$_0$|"BANK","FINANCIAL")
  P("LOAN"$_0$|"BANK","RIVER")
- Intersect several broad semantic domains, and assign a higher weight to the intersected region.

Compare:
P("PETE-ROSE"$_o$|"BASEBALL")
P("PETE-ROSE"$_o$|"GAMBLING")
P("PETE-ROSE"$_o$|"BASEBALL","GAMBLING")

We plan to model multiply triggered sequences in a way that will capture at least some of the above phenomena. This requires statistical analysis of the interaction among the triggers, especially as it relates to the triggered sequence. We have just begun this analysis. One possibility, suggested by Kai-Fu Lee, is to consider the mutual information between the triggers. Triggers with high mutual information provide little additional evidence, and thus should not be added up.

For the system reported below, we considered several simple heuristics: averaging the effect of the different triggers, using the most informative trigger only, and a quickly saturating sum. In the limited context of our current model we found no significant difference between the three.

3.4. Integration with the Static Model

A straightforward way to integrate the trigger model with a static model is to interpolate them linearly, using independent data to determine the weights. A somewhat fancier variant could use weights that depend on the length of the history. We expect the weight of the adaptive component to increase as the history grows. Using linear interpolation, the trigger model can be viewed as an adaptive unigram. This is the solution we used in the system reported below.

However, linear interpolation is not without its faults. Existing static models, such as N-grams, are excellent at using short-range information. For our adaptive component to be useful, it should complement the prediction power of the static component. But linear interpolation means that the adaptive component is blind to short-term constraints, yet the latter strongly affect the behavior of the static model. For example, in processing the sentence

> "The district attorney's office launched an investigation into loans made by several well connected banks."

"DISTRICT-ATTORNEY" may trigger "INVESTIGATION", causing its unigram probability to be raised to its level in documents containing the words "DISTRICT-ATTORNEY". But when "INVESTIGATION" actually occurs, it is preceded by "LAUNCHED AN", which causes a trigram model to predict it with an even higher probability, rendering the adaptive contribution useless.

Thus a better method of combining the two components is to consider the information already provided by the static model. This can be done in two different ways:

- By using a POS-based trigger model, in the spirit of [4].

- By dynamically considering the probabilities produced by the static component, and modifying only those for which the adaptive component provided useful information. We are now experimenting with this method. Since it requires dynamic renormalization, it is only suitable for recognizers which compute the entire array of probabilities for every word.

3.5. The Experiment

We used most of the WSJ LM training corpus, 42M words in all, to train a conventional backoff trigram model[1] for the DARPA 20,000 closed-vocabulary task. We used the same data to derive the triggering list, as described below.

The conditional probability provided by the trigger pair ($A \rightarrow B$) was estimated as:

$$P(B_o|A) = \frac{\text{Count of } B \text{ in documents containing } A}{\text{number of words in documents containing } A} \quad (8)$$

For the unconditional probability $P(B_o)$ we used the static unigram probability of B. We have since switched to using the average probability with which occurrences of B in the training data are predicted by the trigram model, but the results reported here do not reflect this change.

We first created an index of all but the 100 most frequent words, keeping for each word a detailed description of its occurrences. We included paragraph, sentence, and word location information, to allow consideration of different distance measures and different context levels. Excluding the top 100 words reduced the storage requirements by more than 50%. We assumed that frequently used words provide little contextual information. Using the index, we systematically searched for ordered word pairs whose expected utility, as given by Eq. 7, exceeded a given threshold. Of the 400 million possible pairs, we selected some 620,000.

For combining multiple triggering of the same word, we used MAX or AVERAGE or SUM saturating at 2*MAX, as described in section 3.3. We found no significant difference between these methods.

We combined the trigger model with the static trigram using linear interpolation. The automatically derived weights varied from task to task, but were usually in the range of 0.02 to 0.06 for the trigger component. We also tried to use weights that depend on the length of the history, but were surprised to find no improvement.

3.6. Results and Discussion

We tested our combined model on a large collection of test sets, using perplexity reduction as our measure. A selection is given in table 2. Set WSJ-dev is the CSR development test set (70K words). Set BC-3 is the entire Brown Corpus, where the history was flushed arbitrarily every 3 sentences. Set BC-20 is the same as BC-3, but with history-flushing every 20 sentences. Set RM is the 39K words used in training the Resource Management system, with no history flushing. The last result in table 2 was derived by training the trigram on only 1.2M words of WSJ data, and testing on the WSJ development set. This was done to facilitate a more equitable comparison with the results reported in [5].

test set	static PP	dynamic PP	improvement
WSJ-dev	170	153	10%
BC-3	430	311	28%
BC-20	430	293	32%
RM	987	116	88%
WSJ/1.2M-dev	350	295	16%

Table 2: Perplexity reduction by the trigger-based adaptive model for several test sets

Our biggest surprise was that "self triggering" (trigger pairs of the form $(A \rightarrow A)$) was found to play a larger role than would be indicated by our utility measure. Correlations of this type are an important special case, and are already captured by the conventional cache based models. We decided to adapt our model in the face of reality, and maintained a separate self-triggering model that was added as a third interpolation component (the results in table 2 already reflect this change). This independent component, although consisting of far fewer trigger pairs, was responsible for as much as half of the overall perplexity reduction. On tasks with a vastly different unigram behavior, such as the Resource Management data set, the self-triggering component accounted for most of the improvement.

Why do self-triggering pairs have a higher impact than anticipated? One reason could be an inadequacy in our utility measure. Another could spring from the difference between training and testing. If the test set were statistically identical to the training set, the utility of every trigger pair would be exactly as predicted by our expected utility measure. Since in reality the training and testing sets differ, the actual utility is lower than predicted. All trigger pairs suffer a degradation, except for the self-triggering ones. The latter hold their own because self correlations are robust and are better maintained across different corpora. This explains why the self-triggering component is most dominant when the statistical difference between the training and testing data is greatest.

4. SUMMARY AND CONCLUSIONS

We presented two attempts to improve our stochastic language modeling. In the first, we identified a deficiency in the conventional backoff language model, and used statistical reasoning to correct it. Our modified model is about as simple as the original one, but gives a slightly lower perplexity on various tasks. Our analysis suggests that the modification is most suitable when training data is sparse.

In our second attempt, we extended the notion of adaptation to incorporate within-document word sequence correlation, using the framework of a trigger pair. We discussed the issues involved in constructing such a model, and reported promising improvements in perplexity. We have only begun to explore the potential of trigger-based adaptive models. The results reported here are preliminary. We believe we can improve our performance by implementing many of the ideas suggested in sections 3.2, 3.3 and 3.4 above. Work is already under way.

5. ACKNOWLEDGEMENTS

We are grateful to Doug Paul for providing us with the preprocessed CSR language training data in a timely manner; to Dan Julin for much help in systems issues; to Kai-Fu Lee for helpful discussions; to Fil Alleva for many helpful interactions; and to Raj Reddy for support and encouragement.

References

1. Katz, S. M., "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 400–401, March 1987.
2. Jelinek, F., "Self-Organized Language Modeling for Speech Recognition," in *Readings in Speech Recognition*, Alex Waibel and Kai-Fu Lee (Eds.), Morgan Kaufmann, 1989.
3. Kupiec, J., "Probabilistic Models of Short and Long Distance Word Dependencies in Running Text," Proceedings of the Speech and Natural Language DARPA Workshop, pp.290–295, Feb. 1989.
4. Kuhn, R., and De Mori, R., "A Cache-Based Natural Language Model for Speech Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-12, pp. 570–583, June 1990.
5. Jelinek, F., Merialdo, B., Roukos, S., and Strauss, M., "A Dynamic Language Model for Speech Recognition," Proceedings of the Speech and Natural Language DARPA Workshop, pp.293–295, Feb. 1991.
6. Essen, U., and Ney, H., "Statistical Language Modelling Using a Cache Memory," Proceedings of the First Quantitative Linguistics Conference, University of Trier, Germany. September 1991.