PUTTING IT ALL TOGETHER: LANGUAGE MODEL COMBINATION

Joshua T. Goodman

Speech Technology Group
Microsoft Research
One Microsoft Way
Redmond, WA 98052
joshuago@microsoft.com

ABSTRACT

In the past several years, a number of different language modeling improvements over simple trigram models have been found, including caching, higher-order n-grams, skipping, modified Kneser-Ney smoothing, and clustering. While all of these techniques have been studied separately, they have rarely been studied in combination. We find some significant interactions, especially with smoothing techniques. The combination of all techniques leads to up to a 45% perplexity reduction over a Katz smoothed trigram model with no count cutoffs, the highest such perplexity reduction reported.

1. INTRODUCTION

For many years, simple trigram models, smoothed with Katz smoothing [5] or similar techniques, have represented the standard baseline for language modeling. During that time, many improvements over this simple model have been suggested, including skipping [11, 3, 10], clustering [1, 10], caching [6, 7, 8], higher-order n-grams, smoothing [2, 10], and sentence mixture models [4]. While each of these techniques leads to improvements over the baseline, only rarely have the techniques been examined in combination. In many cases, a set of these techniques – for instance, skipping, smoothing and clustering – may all attempt to improve performance in similar ways – in this case, by better combination of data. Similarly, other sets of techniques – e.g. higher-order n-grams and sentence mixture models – may both introduce similar drawbacks, like data sparsity. When a set of techniques has the same advantages or drawbacks, how will they interact? Only a few papers have looked at combinations of techniques, and no paper has tried to combine so many. Putting all of these techniques together leads to perhaps the best reported improvement over a Katz smoothed trigram model, up to a 45% perplexity reduction and an 8.26% reduction in speech recognition word-error rate.

I would like to thank the entire Microsoft Research Speech Technology Group for their help, especially Milind Mahajan, X. D. Huang, and Jianfeng Gao.

2. TECHNIQUE DESCRIPTIONS

In this section, we will describe each of the techniques we have used. We begin by defining some simple notation. The goal of language modeling is to determine the conditional probability of a word, given its history, $P(w_i|w_1...w_{i-1})$. As i becomes large, estimating this probability becomes difficult, so approximations such as $P(w_i|w_1...w_{i-1}) \approx P(w_i|w_{i-2}w_{i-1})$ are typically made. This *trigram* probability is much easier to compute than the fully conditioned probability; a rough approximation is simply $\frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})}$ where $C(w_{i-2}w_{i-1}w_i)$ represents the number of occurences of the sequence $w_{i-2}w_{i-1}w_i$ in some training corpus. Unfortunately, this approximation is noisy, so it is typically *smoothed* by combining it with estimates of $P(w_i|w_{i-1})$ and $P(w_i)$. One of the most commonly used smoothing techniques is called Katz smoothing [5]. We will denote a Katz smoothed trigram model by $P_{\text{Katz}}(w_i|w_{i-2}w_{i-1})$. For many years, Katz smoothing or deleted interpolation represented the state-of-the-art in smoothing techniques.

Smoothing: In [2], a detailed comparison of many smoothing techniques was performed and it was found that a modified interpolated form of Kneser-Ney smoothing [10] consistently outperformed all other smoothing techniques. The basic insight behind Kneser-Ney smoothing is the following. Consider a conventional bigram model of a phrase such as $P_{\text{Katz}}(\textit{Francisco}|\textit{eggplant})$. Since the phrase *San Francisco* is fairly common, the conventional unigram probability (as used by Katz smoothing or deleted interpolation) $\frac{C(\textit{Francisco})}{\sum_w C(w)}$ will also be fairly high. But, the word *Francisco* occurs in exceedingly few contexts, and its probability of occuring in a new one is very low. Kneser-Ney smoothing uses a modified backoff distribution based on the number of contexts each word occurs in, rather than the number of occurrences of the word. Thus, a probability such as $P_{\text{KN}}(\textit{Francisco}|\textit{eggplant})$ would be fairly low, while for a word like *stew* that occurs in many contexts, $P_{\text{KN}}(\textit{stew}|\textit{eggplant})$ would be relatively high, even if the phrase *eggplant stew* did not occur in the training data. In particular, Kneser-Ney smoothing uses the following formula (given here for a bigram)

$$P_{\text{KN}}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)-D}{C(w_{i-1})} + \lambda(w_{i-1})\frac{|\{v|C(vw_i)>0\}|}{\sum_w |\{v|C(vw)>0\}|}$$

where D is a discount factor optimized on heldout data, and $\lambda(w_{i-1})$ is chosen so that the probabilities sum to 1, and where $|\{v|C(vw_i) > 0\}|$ is the number of words v that w_i can occur in the context of. The formula can be easily extended to trigrams and beyond.

Higher-order n-grams: The most obvious extension to trigram models is to simply move to higher-order n-grams, such as four-grams and five-grams. As we will show, when combined with the proper smoothing techniques, five-grams can lead to improved performance. (In pilot experiments, moving beyond five-grams lead to only negligible gains.)

Skipping: Of course, in practice, it may be fairly unlikely that a five-word sequence that occurs in the training data also occurs in the test data; but it will be less unlikely that some similar sequence occurs [11, 3, 10]. In particular, we examine probabilities of the form $P(w_i|w_{i-4}w_{i-3}w_{i-1})$ and $P(w_i|w_{i-4}w_{i-2}w_{i-1})$. There are many possible ways that skipping can be done; the utility of these particular sequences was found during pilot experiments. (For experiments limited to trigrams, we examine probabilities of the form $P(w_i|w_{i-3}w_{i-1})$ and $P(w_i|w_{i-3}w_{i-2})$.)

Clustering: Next, we describe our clustering techniques, which are a bit different (and, in pilot experiments, slightly more effective) than traditional clustering [1, 10]. Consider a probability such as $P(\textit{Tuesday}|\textit{party on})$. Perhaps the training data contains no instances of the phrase *party on Tuesday*, although other phrases such as *party on Friday* do appear. If we put words into classes, such as the word *Tuesday* into the class *WEEKDAY*, we can then decompose the probability

$$P(\textit{Tuesday}|\textit{party on}) = P(\textit{WEEKDAY}|\textit{party on}) \times P(\textit{Tuesday}|\textit{party on WEEKDAY})$$

This decompostion is a strict equality, but in practice, because of smoothing, may lead to significantly better results. More generally, we will denote this probability as

$$P(W_i|w_{i-2}w_{i-1}) \times P(w_i|w_{i-2}w_{i-1}W_i)$$

We call this technique "predictive" clustering.

Another way of using clustering is on the conditioned words. For instance, if parties are in the class of *EVENT*, we could compute $P(\textit{Tuesday}|\textit{EVENT on})$ If there were no occurrences in the training data of *EVENT on Tuesday* we could then backoff to $P(\textit{Tuesday}|\textit{on})$; if there were no occurrences of *on Tuesday* we could back off to $P(\textit{Tuesday}|\textit{PREPOSITION})$ before finally backing off to the unigram. As a concise way of indicating this backoff, we will write

$$P(\textit{Tuesday}|\textit{PREPOSITION on EVENT party})$$

More generally, we will denote this kind of backoff by $P(w_i|w_{i-2}W_{i-2}w_{i-1}W_{i-1})$, to mean the backoff order is from $P(w_i|w_{i-2}W_{i-2}w_{i-1}W_{i-1})$ to $P(w_i|W_{i-2}w_{i-1}W_{i-1})$ to $P(w_i|w_{i-1}W_{i-1})$ to $P(w_i|W_{i-1})$ to $P(w_i)$. Since we always use hard clusters – clusters in which each word belongs to a single cluster – $P(w_i|w_{i-2}W_{i-2}w_{i-1}W_{i-1}) = P(w_i|w_{i-2}w_{i-1})$, and similarly for the other probabilities. We call this technique "conditional" clustering.

These techniques can be combined together to produce even better results than either could produce separately. In particular, we use the following formula:

$$P(w_i|w_{i-2}w_{i-1}) = P(W_i|w_{i-2}W_{i-2}w_{i-1}W_{i-1}) \times P(w_i|w_{i-2}W_{i-2}w_{i-1}W_{i-1}W_i)$$

There is no need for the clusters used for the predictive clustering and the conditional clustering to be the same [12]. For instance, consider a pair of words like *a* and *an*. In general, *a* and *an* can follow the same words, and so, for predictive clustering, belong in the same cluster. But, there are very few words that can follow both *a* and *an* – so for conditional clustering, they belong in different clusters. We have also found in pilot experiments that the optimal number of clusters used for predictive and conditional clustering are different; in this paper, we use 128 clusters for the conditional clusters, and 256 clusters for the predictive clusters.

The clusters are found automatically using a tool that attempts to minimize perplexity. In particular, for the conditional clusters, we try to minimize the perplexity of training data for a bigram of the form $P(w_i|W_{i-1})$; for the conditional clusters, we try to minimize $P(w_{i-1}|W_i)$ – although this is not quite the correct quantity to minimize, it means that we can use the same clustering code with minimal modifaction (switch the order of the bigrams) for both kinds of clustering, and it works well in practice.

Caching: If a speaker uses a word, it is likely that he will use the same word again in the near future. This observation is the basis of caching [6, 7, 8]. In particular, in a unigram cache, we form a unigram model from the most recently spoken words (all those in the same article if article markers are available, or a fixed number of previous words if not.) This unigram cache can then be linearly interpolated with a conventional n-gram. We can use other kinds of caches; for instance, we could form a smoothed trigram from the previous words, and interpolate. The best performing technique we have tried – the one used here in experiments – is to use a unigram cache, and a smoothed trigram cache; however, the smoothed trigram cache is only used if either the bigram or trigram context of the data is found somewhere in the cache. The weights for both caches vary linearly with the amount of data in the cache. This technique was marginally better than a straight unigram interpolation in pilot experiments.

Sentence Mixture Models: Iyer and Ostendorf [4] observed that within a corpus, there may be several different sentence types, and that by modeling each sentence type separately, improved performance can be achieved. In particular, let s_j denote the condition that the sentence under consideration is a sentence of type j. Then the probability of the sentence, given that it is of type j can be written as

$$\prod_{i=1}^{n} P(w_i|w_{i-2}w_{i-1}s_j)$$

Sometimes, the global model (across all sentence types) will be better than any individual sentence type. Let s_0 be a special context that is always true:

$$P(w_i|w_{i-2}w_{i-1}s_0) = P(w_i|w_{i-2}w_{i-1})$$

Let there be S different sentence types ($4 \leq S \leq 8$ is typical); let $\sigma_0 ... \sigma_S$ be sentence interpolation parameters optimized on held-out data subject to the constraint $\sum_{j=0}^{S} \sigma_j = 1$. The overall probability of a sentence can be denoted by

$$\sum_{j=0}^{S} \sigma_j \prod_{i=1}^{n} P(w_i | w_{i-2} w_{i-1} s_j)$$

The probabilities $P(w_i|w_{i-2}w_{i-1}s_j)$ may suffer from data sparsity, so they are linearly interpolated with the global model $P(w_i|w_{i-2}w_{i-1})$. This formulation is equivalent to one in which the sentence type is a hidden variable for each test sentence. Sentence types for the training data were found by using the same clustering program used for clustering words; in this case, we tried to minimize the sentence-cluster unigram perplexities, a slighly simpler technique than that previously used [4].

Combining techniques: Our overall technique is somewhat complicated. At the highest level, we use a sentence mixture model, in which we sum over sentence-specific models for each sentence type. Within a particular sentence mixture model, we combine different techniques with predictive clustering. That is, we combine sentence-specific, global, cache, and global skipping models first to predict the cluster of the next word, and then again to predict the word itself given the cluster. Our overall technique can be summarized by the following equations.

For each sentence type, we wish to linearly interpolate the sentence-specific 5-gram model with the general 5-gram model, the two skipping models, and the two cache models. We wish to do this whether we are predicting the cluster of the word given its context, or the word, given both cluster and context. Let $\lambda_{1,j} ... \lambda_{6,j}$ be interpolation parameters (optimized on held-out data). Then, the function $sen_j(A, w_{i-4}...w_{i-1}, B)$ is

$$\begin{aligned}
&sen_j(A, w_{i-4}...w_{i-1}, B) = \\
&\lambda_{1,j} P_{\text{KN}}(A|w_{i-4}W_{i-4}w_{i-3}W_{i-3}w_{i-2}W_{i-2}w_{i-1}W_{i-1}Bs_j) + \\
&\lambda_{2,j} P_{\text{KN}}(A|w_{i-4}W_{i-4}w_{i-3}W_{i-3}w_{i-2}W_{i-2}w_{i-1}W_{i-1}B) + \\
&\lambda_{3,j} P_{\text{KN}}(A|w_{i-4}W_{i-4}w_{i-3}W_{i-3}w_{i-1}W_{i-1}B) + \\
&\lambda_{4,j} P_{\text{KN}}(A|w_{i-4}W_{i-4}w_{i-2}W_{i-2}w_{i-1}W_{i-1}B) + \\
&\lambda_{5,j} P_{\text{cache}}(A|B) + \\
&\lambda_{6,j} P_{\text{cache}}(A|w_{i-2}w_{i-1}B)
\end{aligned}$$

A and B will either be the class of the word we are trying to predict and a dummy value ($\bullet$), respectively, or the word we are trying to predict and the class of the word.[1]

Now, we can write out our probability model:

$$P_{\text{everything}}(w_1...w_n) = \sum_{j=0}^{S} \sigma_j \prod_{i=1}^{n} sen_j(W_i, w_{i-4}...w_{i-1}, \bullet) \times sen_j(w_i, w_{i-4}...w_{i-1}, W_i)$$

The parameters used for Kneser-Ney smoothing, as well as $\lambda_{k,j}$ are actually different, depending on whether we are predicting a cluster or a word.

[1]This formula is actual an oversimplification because the values $\lambda_{5,j}$ and $\lambda_{6,j}$ depend on the amount of training data in a linear fashion, and if the context $w_{i-1}B$ does not occur in the cache, then the trigram cache is not used. In either case, the values of the λ's have to be renormalized so that they sum to 1.

Clearly, combining all of these techniques together is not easy, but as we will show in the results section, the effects of combination are roughly additive, and the effort is worthwhile.

3. RESULTS

We performed several sets of experiments. We primarily performed our experiments on data with verbalized punctuation, but the data we used for acoustic testing used non-verbalized punctuation. All of our experiments used the NAB corpus, primarily the Wall Street Journal (WSJ) section. In particular, we always used ws94_044 for heldout data, and ws94_045 for test. Because of the way our code was implemented, we used relatively small heldout and test sets consisting of every eighth sentence taken from the first 160,000 words of the respective files. We performed tests using 100,000, 1,000,000, 10,000,000 words of training data (all taken from the beginning of the WSJ section of NAB), and the complete corpus (except heldout and test): 290 million words for verbalized punctuation, 260 million words for non-verbalized punctuation. End-of-sentence was included in perplexity computations, but out-of-vocabulary words were not. The vocabulary size was always 60,000 words.

In Figure 1 we display perplexity results. The results are presented in a slightly unusual way: since it was unclear what order to add the techniques in, we decided to show results from a baseline, with each technique added or removed individually, rather than incrementally. Thus, for instance, a chart entry in the column "Katz+" and the row "sentence" indicates Katz smoothing combined with sentence clustering; an entry in the column "All-" and row "cluster" indicates all techniques except clustering. Because of interactions between some techniques and smoothing methods, we also used Kneser-Ney smoothing as a baseline.

Examining the results, we find some trends that are expected, and others that are more surprising. Perhaps most interesting is the interaction between clustering and smoothing, which had not been observed (or tested for) previously. We also note a relatively small but consistent interaction between sentence mixture models and training size: the larger the training size, the more the improvement from sentence mixture models. Predictably, the effect of caching is much smaller with larger training data sizes. Improvements from skipping models grow with the training data size, but are negligible when the best other techniques are used.

We also performed word-error rate experiments rescoring 100-best lists of WSJ94 dev and eval, about 600 utterances. The 1-best error rate for the 100-best lists was 10.1% (our recognizer's models were slightly worse than even the baseline used in rescoring) and the 100-best error rate (minimum possible from rescoring) was 5.2%. We were not able to get word-error rate improvements by using caching (when the cache consisted of the output of the recognizer), and were actually hurt by the use of caching when the interpolation parameters were estimated on correct histories, rather than on recognized histories. The bottom right chart of Figure 1 shows word-error rate improvement of each technique, from three baselines: Katz, Kneser-Ney,

	100,000			1,000,000			10,000,000		
	Katz+	KN+	All-	Katz+	KN+	All-	Katz+	KN+	All-
perplexity	404.07	358.75	235.56	221.51	196.97	126.51	136.28	124.95	77.09
%improve			41.70%			42.89%			43.43%
skip	6.61%	4.72%	1.34%	8.09%	5.73%	2.45%	9.16%	7.14%	2.23%
5-gram	-5.62%	0.70%	-2.47%	-7.77%	1.31%	-1.35%	-5.92%	5.18%	2.17%
sentence	6.64%	5.61%	6.57%	7.19%	6.13%	5.17%	8.10%	7.65%	4.09%
cluster	-33.59%	-36.10%	-5.79%	-15.08%	-6.78%	5.07%	-6.10%	5.50%	11.00%
cache	36.88%	35.10%	66.10%	29.65%	28.09%	44.06%	25.15%	24.16%	30.45%
KN	11.22%		15.07%	11.08%		21.02%	8.31%		23.38%

	all			all-no-punc			Error rates – all-no-punc		
	Katz+	KN+	All-	Katz+	KN+	All-	Katz+	KN+	All-cache-
perplexity	66.14	63.83	36.37	95.20	91.47	55.74	90.31	90.40	91.11
%improve			45.02%			41.45%			8.26%
skip	9.55%	9.26%	3.88%	11.16%	10.53%	5.40%	1.03%	2.40%	1.24%
5-gram	14.70%	22.66%	23.15%	13.89%	22.12%	22.79%	-0.52%	2.81%	1.46%
sentence	9.21%	9.43%	7.08%	9.39%	9.43%	7.79%	1.55%	0.73%	1.35%
cluster	-0.36%	4.17%	8.79%	-2.75%	4.56%	10.26%	1.55%	3.44%	4.50%
cache	15.07%	14.77%	13.91%	7.77%	7.45%	6.21%	-2.99%	-1.35%	
KN	3.49%		22.66%	3.91%		27.80%	0.93%		7.54%

Figure 1: Relative improvement from each technique on a variety of sizes

and everything, except caching. The most important single factor for word-error rate was the use of Kneser-Ney smoothing, which leads to a small gain by itself, but also makes clustering, skipping, and 5-grams much more effective. Clustering also leads to significant gains.

4. CONCLUSION

We believe our results – up to a 45% perplexity reduction – are the best ever reported for language modeling, as measured by improvement from a fair baseline, a Katz smoothed trigram model with no count cutoffs. Our word-error rate reduction of 8.26% from combining all techniques except clustering is also very good. The results compare favorably to other recently reported combination results [9], where, essentially using a subset of these techniques, from a comparable baseline (absolute discounting), the perplexity reduction is half as much. Our results show that smoothing can be the most important factor in n-gram modeling, and its interaction with other techniques cannot be ignored.

5. REFERENCES

[1] P. F. Brown, V. J. DellaPietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December 1992.

[2] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, October 1999.

[3] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld. The SPHINX-II speech recognition system: An overview. *Computer, Speech, and Language*, 2:137–148, 1993.

[4] R. Iyer and M. Ostendorf. Modeling long distance dependence in language: Topic mixtures versus dynamic cache models. *IEEE Transactions on Acoustics, Speech and Audio Processing*, 7:30–39, January 1999.

[5] S. M. Katz. Estimation of probabilities from sparse data for the langauge model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(3):400–401, March 1987.

[6] R. Kuhn. Speech recognition and the frequency of recently used words: A modified markov model for natural language. In *12th International Conference on Computational Linguistics*, pages 348–350, Budapest, August 1988.

[7] R. Kuhn and R. D. Mori. A cache-based natural language model for speech reproduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583, 1990.

[8] R. Kuhn and R. D. Mori. *Correction to* a cache-based natural language model for speech reproduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):691–692, 1992.

[9] S. Martin, C. Hamacher, J. Liermann, F. Wessel, and H. Ney. Assessment of smoothing methods and complex stochastic language modeling. In *6th European Conference on Speech Communication and Technology*, volume 5, pages 1939–1942, Budapest, Hungary, September 1999.

[10] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer, Speech, and Language*, 8:1–38, 1994.

[11] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, April 1994.

[12] H. Yamamoto and Y. Sagisaka. Multi-class composite n-gram based on connection direction. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Phoenix, Arizona, May 1999.