Scalable Trigram Backoff Language Models

Kristie Seymore Ronald Rosenfeld

May 1996

CMU-CS-96-139

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship and the Department of the Navy, Naval Research Laboratory under Grant No. N00014-93-1-2005.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government or the National Science Foundation.

Keywords: speech recognition, statistical language modeling, scalable language models

Abstract

When a trigram backoff language model is created from a large body of text, trigrams and bigrams that occur few times in the training text are often excluded from the model in order to decrease the model size. Generally, the elimination of n-grams with very low counts is believed to not significantly affect model performance. This project investigates the degradation of a trigram backoff model's perplexity and word error rates as bigram and trigram cutoffs are increased. The advantage of reduction in model size is compared to the increase in word error rate and perplexity scores.

More importantly, this project also investigates alternative ways of excluding bigrams and trigrams from a backoff language model, using criteria other than the number of times an n-gram occurred in the training text. Specifically, a difference method has been investigated where the difference in the logs of the original and backed off trigram and bigram probabilities was used as a basis for n-gram exclusion from the model. We have shown that excluding trigrams and bigrams based on a weighted version of this difference method results in better perplexity and word error rate performance than excluding trigrams and bigrams based on counts alone.

1 Introduction

A language model is a fundamental component of a speech recognizer that assigns prior probabilities to hypothesized word sequences supplied by a speech decoder. Statistical language models estimate these prior probabilities by counting the number of occurrences of all words and certain word sequences of interest in a given training text. The reliability of these probability estimates when used for a particular speech recognition task depend on the source of the training text and the amount of training text available. Inaccuracies in probability estimates frequently arise when the training text does not resemble the nature of the language to be recognized, or when there is not enough training data available to obtain a reliable representation of word and word sequence frequency from a particular text source.

Current collections of text for statistical language model training are making the sparse training data problem less serious for certain domains, such as ARPA's Wall Street Journal corpus, which is part of the 305 million word North American Business News collection. The more training text that is used for language model creation, the more unique word sequences are encountered that must be stored in the model. Thus, as training text size increases, language model size necessarily increases, which can lead to models that are too unwieldy and memory-demanding to be of practical use. This overabundance of training data will allow us, or more correctly force us, to be selective in choosing the training data that we use to create our models.

This project investigates methods of training text pruning that allow for compact and efficient creation of trigram backoff language models. One pruning method, the popular cutoff method, eliminates from the trigram backoff model those bigrams and trigrams that occur the fewest number of times in the training text. We develop another method based on weighted differences, where the difference in the logs of the original and backed off trigram and bigram probabilities is used as a basis for n-gram exclusion from the model. Perplexity and word error rates are used to assess a model's performance as fewer bigrams and trigrams are incorporated into the model. Also, different amounts of training text are pruned down to create models of the same size, so that the effect of the original amount of training data on a scaled-down model can be concluded. These results help determine if the statistical advantages of creating a language model from a large training text can be carried over to scaled down versions of the same model, for use on systems whose memory capacities do not meet the large model's requirements.

2 The Backoff Language Model

The backoff language model was developed by Katz [2] to address the problems associated with sparse training data. Small amounts of training data are more likely to misrepresent the true distribution of word frequencies from a particular language source due to a lack of sufficient samples. The backoff model handles this type of sampling error by reducing the probability of unreliable estimates made from observed frequencies and distributing this freed probability mass among those words from a given vocabulary that did not occur in the training text [2]. Generally, an estimate is deemed unreliable if it occurred few times in the training text. Word sequences, or n-grams, with low counts have their maximum-likelihood estimates replaced by Turing's estimates.

The trigram backoff model is constructed by counting the frequency of unigrams, bigrams and trigrams in a sample text relative to a given vocabulary. Those n-grams that occur few times in the text are discounted, and the extra probability mass is divided among those words in the vocabulary that are not seen in the training data. As a result, every word in the vocabulary has a finite probability of occurring when the model is used to predict new word sequences. The model also tries to use as much word history as possible when assigning the probability of a word given the two words that precede it. The probability assigned to a word sequence is shown in Equation 1, where w_j^k represents the sequence $(w_j, \ldots w_k)$, the discount ratio d is a function of the count $C(w_1^n)$, and the α's are the backoff weights that ensure that the probabilities sum to one:

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

In the case of a trigram model, the word history is limited to the two words preceding the word for which we are defining a probability. The model tries to assign the trigram probability $P(w_n|w_{n-2}^{n-1})$ if it exists in the model. If there is no trigram probability for that word sequence, then the model backs off and uses a weighted version of the bigram probability, $P(w_n|w_{n-1})$. If the bigram was not seen in the training text, the model backs off again and uses a weighted version of unigram probability $P(w_n)$.

As the amount of training text used to create the backoff model increases, the number of unique trigrams and bigrams increases. (The number of unigrams stays constant and equals the size of the vocabulary.) The language model will necessarily takes up more memory in order to store the additional information from the training text. At some point, the model's memory requirements will exceed any practical system capacity. Therefore, we can either limit the amount of training data we use to develop the model, or take from a large amount of training text that portion which leads to the most reliable word predictions. Thus, a large amount of training text can be scaled down to create a compact model of a desired size. Two scaling techniques that can be used to select specific amounts of training text will be introduced in the following sections.

3 Pruning Techniques

As has been previously suggested, word sequences that occur the fewest number of times in a training text can lead to unreliable predictions. This idea has led to the popular cutoff method of training text reduction, where only information about the most frequently occurring bigrams and trigrams is included in the language model. This method will be explored in depth in Section 3.1. However, we also need to consider the word sequences for which the model would not make a good prediction if the probability for that sequence was not included in the model. There may be a way to exploit the redundancy present in the structure of the backoff model by specifically considering which information from the training data would do the most harm if excluded. This idea has led to the development of the weighted difference method of training text reduction, which will be introduced in Section 3.2.

3.1 The Cutoff Method

The cutoff method of training text pruning excludes from the language model those bigrams and trigrams that occur infrequently. The motivation for this method lies in the argument that there is not much difference between a trigram or bigram occurring once in a text of millions of words and it not occuring at all. A trigram will occur only once in a training text for one of two reasons: either the trigram represents a word sequence that is rarely emitted from the training text source, or a sampling error has occurred and the training text does not accurately reflect the true expected frequency of that trigram. In either case, there should not be much harm done to the model's performance if that trigram is excluded from the model. If the trigram does in fact occur only rarely, then it will most likely occur infrequently during the model's use, leading to infrequent probability errors. The model will back off to a bigram probability estimate, which will hopefully lead to a stronger (more reliable) probability estimate for those cases where a sampling error has occurred. Just by excluding those trigrams with a count of one from a model, a significant savings in memory can be achieved. In a typical training text, roughly 80% of unique trigram sequences occur only once. This idea can be extended further to bigrams and trigrams that occur more than once. We can designate a trigram cutoff and a bigram cutoff, where a cutoff of k means that n-grams occuring k or fewer times are discarded.

What kind of memory savings can we expect from excluding bigrams and trigrams in this manner? In Carnegie Mellon University's Sphinx II speech recognizer, each trigram takes up 4 bytes of memory (2 bytes for word identification and 2 bytes for the probability) and each bigram takes 8 bytes (2 bytes for word identification, 2 bytes for the probability, 2 bytes for the backoff weight and 2 bytes for a pointer to the trigrams.) The memory required for unigram probabilities and constants can be considered a fixed overhead, and is not included in our memory calculations. Using a 58,000 word dictionary and 45 million words of Wall Street Journal training data (1992 - 1994), the memory requirements of models created with different cutoffs can be computed. Several sample model sizes are shown in Table 1, with cutoffs indicated by (bigram cutoff – trigram cutoff).

Model Cutoffs	# Bigrams	# Trigrams	Memory (MB)
(0–0)	4,627,551	16,838,937	104
(0–1)	4,627,551	3,581,187	51
(1–1)	1,787,935	3,581,187	29
(0–10)	4,627,551	367,928	38
(10–10)	347,647	367,928	4

Table 1: Model Cutoffs and Resulting Model Size

For this data, 78.5% of the trigrams and 61% of the bigrams occur only once, so we see that significant memory savings can be obtained by cutting out the bigrams and trigrams that appear infrequently.

In order to investigate the effects of raising bigram and trigram cutoffs, several models were created using the Carnegie Mellon Statistical Language Modeling Toolkit [4]. The perplexities of the scaled down models were computed using the official ARPA 1994 Language Model Development Set, and the word error

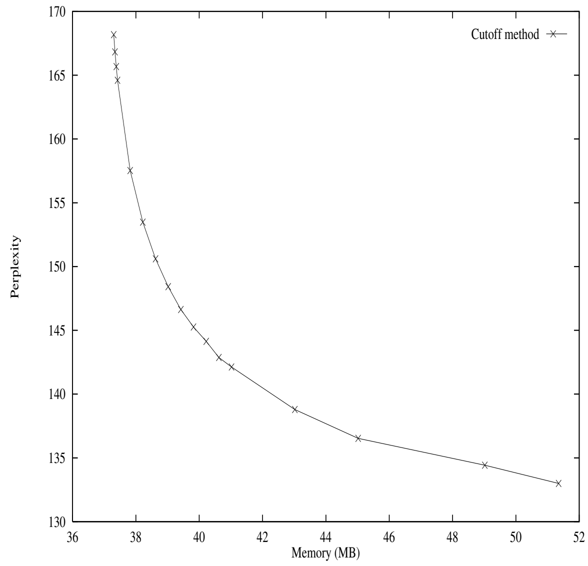

Figure 1: Perplexity vs Scaled Language Model Size, Trigram Pruning Only, 1992 - 1994 Data.

rate was computed using CMU's Sphinx II system and the ARPA 1994 Hub 1 Acoustic Development Set (7387 words). The number of trigrams to be retained in the model was chosen to be some fixed number, and then the cutoff was set to be the maximum cutoff possible so that all trigrams with a count equal or less than the cutoff plus some number with a count of (cutoff+1) were removed from the model. The trigrams cut out at level (cutoff+1) were the first ones encountered in an alphabetized list.

Figures 1 and 2 show the effects of cutting out only trigrams on perplexity and word error rate. Figures 3 and 4 show the effects of cutting out both bigrams and trigrams from the model. For each model, a fixed number of trigrams was chosen to be retained in the model. For bigram pruning, the number of bigrams retained in the model was as close as possible to the number of trigrams in the model. The bigram and trigram cutoffs were chosen so that these desired totals could be met, resulting in bigram and trigram cutoffs that were not necessarily the same number. As can be seen from the graphs, the savings in terms of memory are much greater when bigrams are excluded from the model.

As the language model size is decreased, the perplexity rises sharply (Figures 1 and 3.) The word error rate also rises sharply, although the increase only truly

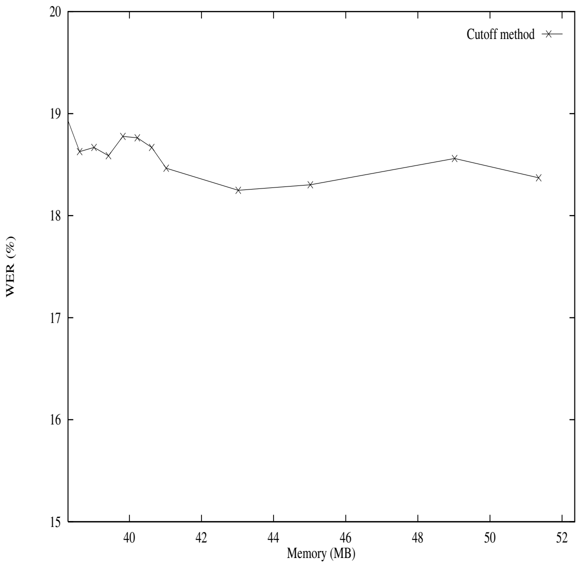

Figure 2: Word Error Rate vs Scaled Language Model Size, Trigram Pruning Only, 1992 - 1994 Data.

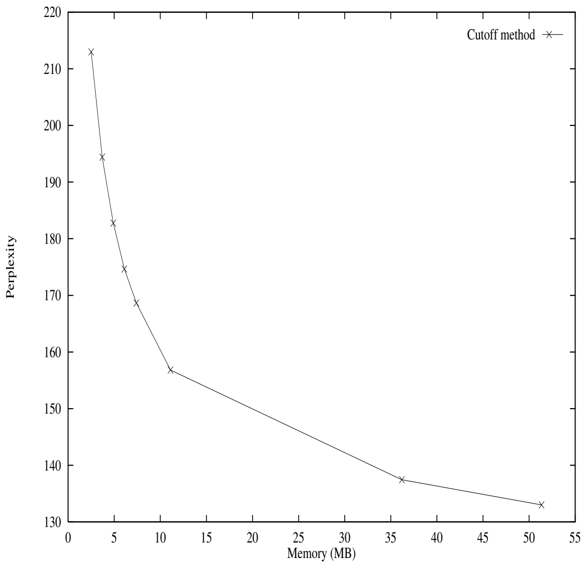

Figure 3: Perplexity vs Scaled Language Model Size, Bigram and Trigram Pruning, 1992 - 1994 Data.

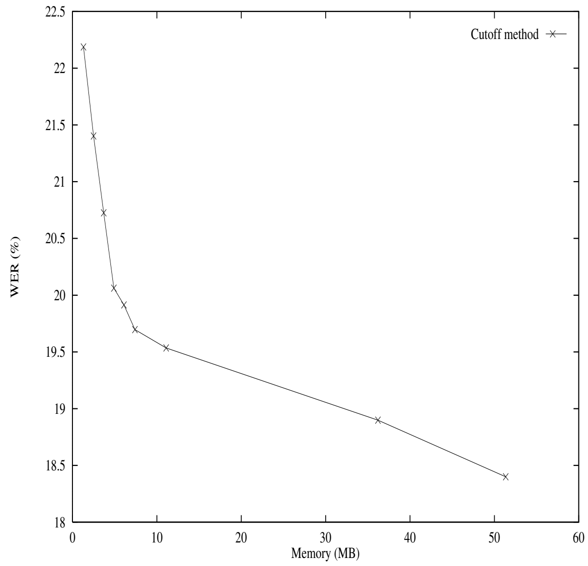

Figure 4: Word Error Rate vs Scaled Language Model Size, Bigram and Trigram Pruning, 1992 - 1994 Data.

becomes significant once bigrams are pruned from the model.

3.2 The Weighted Difference Method

The cutoff method shows that training text reduction can result in a significant savings in memory with a generally acceptable increase in WER and perplexity. However, there should be a more intelligent way to choose which trigrams and bigrams to include in the model than just those which occur the most often in the training text. If an n-gram is not present in the model, the model uses a backed off probability estimate in place of the original estimate. If that backed off estimate is very close to the original estimate, then there is no need to store the original estimate in the first place. This idea has led to the weighted difference method of training text reduction.

The weighted difference factor of an n-gram is defined to be

$$w.d.factor = K * (\log(\text{original prob}) - \log(\text{backedoff prob})) \qquad (2)$$

where K is the Good-Turing discounted n-gram count. This factor reflects our desire to keep an n_gram in the language model.

The CMU Statistical Language Modeling Toolkit was modified to create weighted difference language models by pruning n-grams based on their weighted difference factor. Several models were created. The results are plotted with the cutoff method results, and are shown in Figures 5 - 8. In both cases, as the language model size is decreased, the perplexity rises sharply. Trigram pruning does not have much effect on WER, but bigram pruning causes memory savings and increases in WER to become significant.

As can be seen from these figures, the models created with the weighted difference method have significantly lower perplexity values than for the cutoff models, but the perplexity rises in the same manner in both cases. The word error rates for the weighted difference models are almost always lower than that of the cutoff models, but the significance of the difference is questionable. We can say with confidence that using the weighted difference method is at least as good as the cutoff method, and generally yields improved perplexity and word error rates over the cutoff method.

Table 2 displays more clearly the results depicted in Figure 8 for the weighted difference method, with the relative increase in WER over the original (0–1) model shown.

# Bigrams	# Trigrams	Memory (MB)	WER (increase)
4,627,551	3,581,187	51 MB	(original model)
4,627,551	400,000	39 MB	1% relative
4,627,551	70,000	37 MB	3% relative
934,351	900,000	11 MB	5% relative
416,338	400,000	5 MB	9% relative
108,117	100,000	1.3 MB	20% relative

Table 2: Model Reduction and Resulting WER Increases

Is model size reduction a feasible practice? We see in Table 2 that significant memory reduction can be achieved. Certainly, for particular applications, the increase in WER is worth the savings in memory.

4 Effects of Different Amounts of Training Data

In order to verify that using more training data and then pruning it down is a better approach than just starting with a smaller body of training data, three different

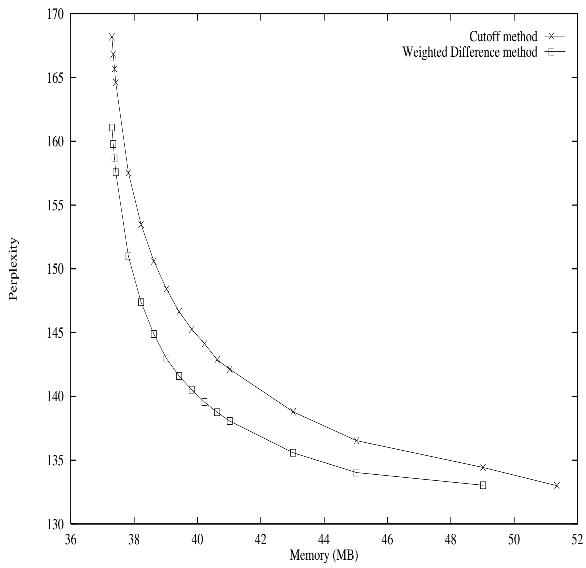

Figure 5: Perplexity vs Scaled Language Model Size, Trigram Pruning Only, 1992 - 1994 Data.

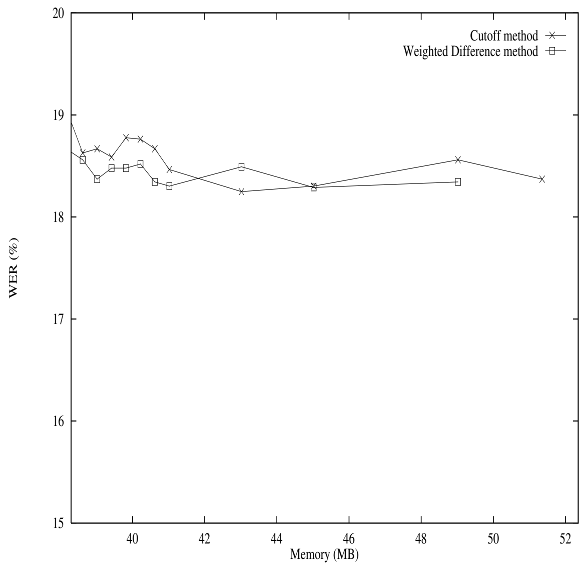

Figure 6: Word Error Rate vs Scaled Language Model Size, Trigram Pruning Only, 1992 - 1994 Data.

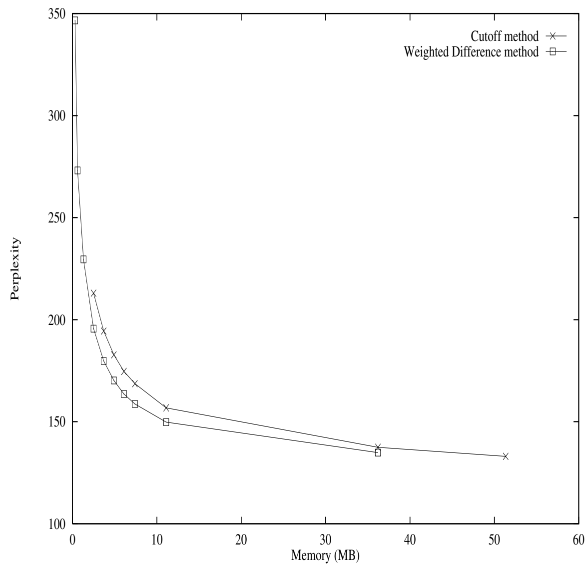

Figure 7: Perplexity vs Scaled Language Model Size, Bigram and Trigram Pruning, 1992 - 1994 Data.

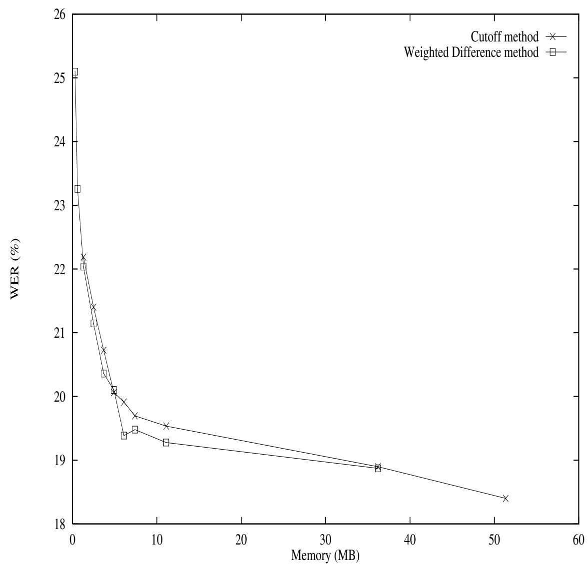

Figure 8: Word Error Rate vs Scaled Language Model Size, Bigram and Trigram Pruning, 1992 - 1994 Data.

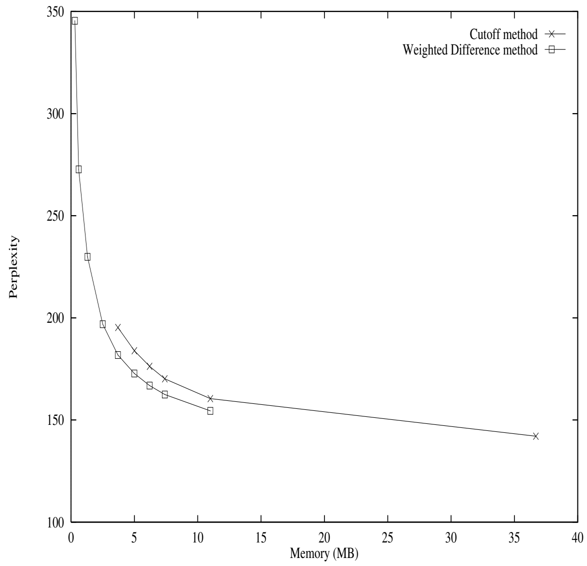

Figure 9: Perplexity vs Scaled Language Model Size, Bigram and Trigram Pruning, 1993 - 1994 Data.

sized data sets were defined and used to create models of the same size. The first data set consists of 45.3 million words of Wall Street Journal data (1992 - 1994), the same data set used in the examples above. The second data set is a subset of the first data set, consisting of 28.5 million words of Wall Street Journal data from 1993 - 1994. The third set is yet a smaller set, 6.5 million words of 1994 Wall Street Journal data. Several language models of approximately the same size were computed with the three data sets using both the cutoff and weighted difference methods, pruning as many bigrams and trigrams as necessary in order to reach the desired size. The perplexity and word error rate results are shown in Figures 9 - 12 for the second (1993 - 1994) and the third (1994 only) data sets. Refer to Figures 7 and 8 for the 1992 - 1994 results. For the third set of data (6.5 million words), the largest memory data point represents a (0–0) model, where no pruning has occurred at all. For all three sets, the weighted difference method generally outperforms the cutoff method in terms of perplexity and word error rate.

Figures 13 and 14 show the weighted difference results for all three data sets. It can clearly be seen that the 6.5 million word models perform significantly worse than the models originally created from 45.3 and 28.5 million words. The difference between the first and second data sets is not as significant, yet the larger

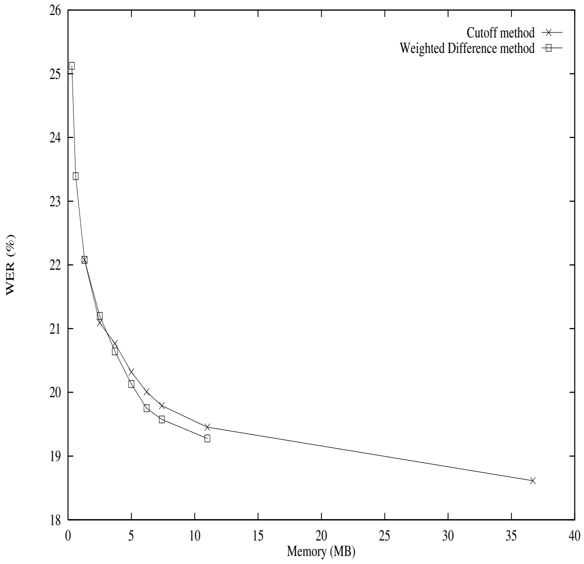

Figure 10: Word Error Rate vs Scaled Language Model Size, Bigram and Trigram Pruning, 1993 - 1994 Data.

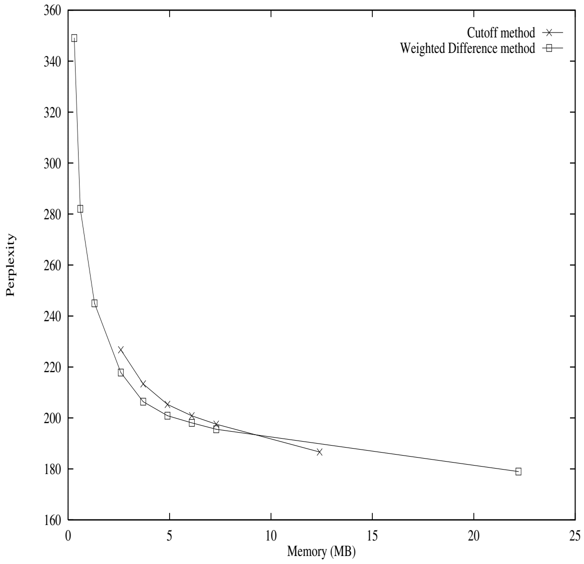

Figure 11: Perplexity vs Scaled Language Model Size, Bigram and Trigram Pruning, 1994 Data.

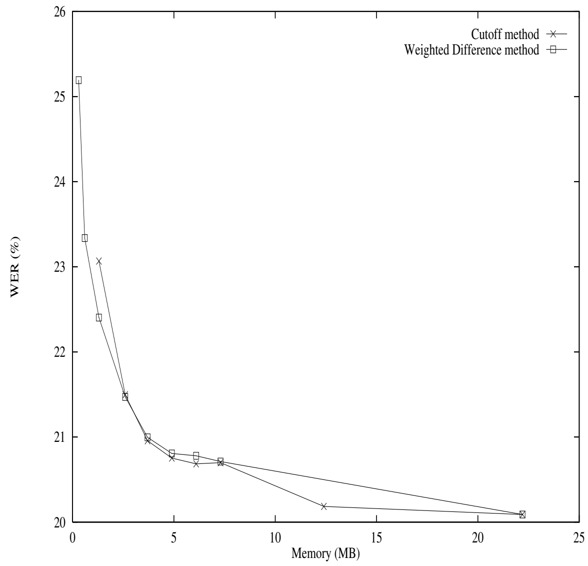

Figure 12: Word Error Rate vs Scaled Language Model Size, Bigram and Trigram Pruning, 1994 Data.

data set does do slightly better.

There are several factors that need to be considered when analyzing the results of Figures 13 and 14. First of all, the three data sets do not come from the same distribution. There is a time shift present, in that the data that is added to the 6.5 million words to get the 28.5 and 45.3 million words is older data. If a significant change of style or content has occurred over time for that source, the statistics of the older data may be less helpful in modeling probabilities due to bigram and trigram frequencies that do not accurately reflect the current frequency distributions of the language source. In fact, we found a consistent 10% perplexity increase when the 6.5 MW of 1994 data was replaced by a comparable amount of 1992 data. In previous work ([3]), we found a similar effect on the OOV rate.

Second, there seems to be a threshold at about the 1–2 MB model size for which the perplexities and word error rates degrade equally no matter how much data was used initially. At some point, so much information has been pruned from the model that perhaps the models converge to approximately the same set of bigram and trigram sequences, which are those that occur the most frequently. For example, 92% of the bigrams and 87% of the trigrams are the same in the two 1.3 MB models based on 28.5 MW and 45.3 MW. Further intersecting with the 6.5 MW

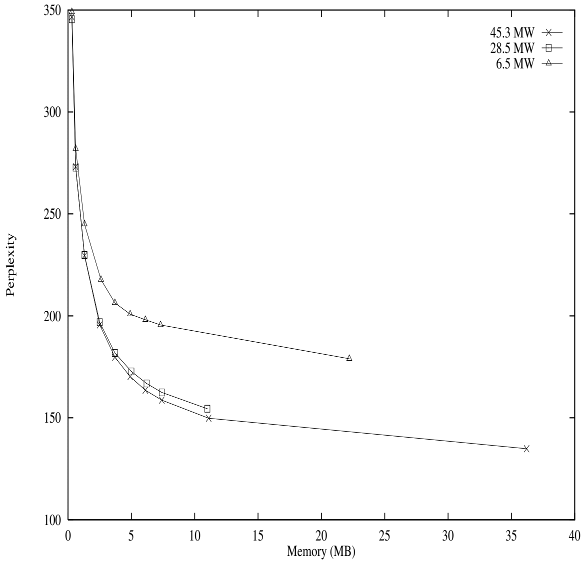

Figure 13: Perplexity vs Scaled Language Model Size for Different Amounts of Training Data (Weighted difference pruning.)

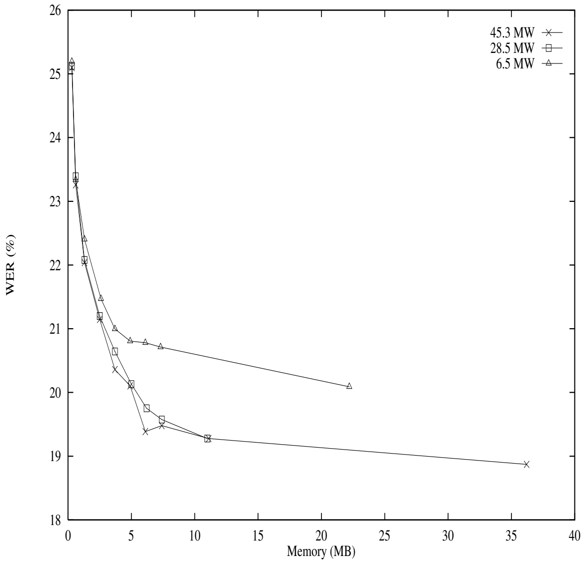

Figure 14: Word Error Rate vs Scaled Language Model Size for Different Amounts of Training Data (Weighted difference pruning.)

model yields a 73% bigram and a 59% trigram overlap. Using approximately the same set of bigrams and trigrams with approximately the same set of probabilities is likely to lead to similar performance.

5 Conclusions

From the results presented in the previous sections, we can come to the following conclusions about the usefulness and performance of compact language models:

- Training text pruning can be used to build compact and efficient language models that require significantly less memory than language models built from complete training text.
- As model size decreases, the weighted difference method of training text pruning results in a significantly smaller perplexity increase than the cutoff method.
- As model size decreases, the weighted difference method of training text pruning generally results in a slightly smaller word error rate increase than the cutoff method.
- Using more training data, up to at least 25 - 30 million words initially, and then pruning it down is a better approach than just starting with a small amount of training data, as long as the training text does not contain significant style changes and the pruning is not severe (at least 2MB remaining). Beyond 25 million words, the amount of training data does not have a noticeable effect.

6 Future Work

There are several issues of interest that could be addressed in future work on scalable language models. Currently, the weighted difference method provides two lists: one for bigrams and one for trigrams. The user decides how many trigrams and bigrams to include in the scaled down model. A one list approach would be much more useful, where the system decides automatically whether a trigram or bigram should be the next entity pruned from the model. The two list approach was used here because the memory requirements of bigrams and trigrams are implementation dependent, and also due to the complication of trigram dependencies on bigrams. In our implementation of the backoff language model, a bigram cannot be excluded from the model if it is an initial part of a trigram that is still in the model. Trigram pruning occurs first, and then the bigrams are pruned only if they have no dependent trigrams. A different implementation of the backoff model may be able to avoid this problem.

Alternatively, a one list approach can be used despite trigram dependencies on bigrams. One list can be made containing all of the weighted difference factors for all of the bigrams and trigrams in a large model, and then the top n-grams from the list (those with the highest weighted difference factors) can be chosen in order to meet a desired memory size. Whenever a bigram is to be excluded from the model, all of its dependent trigrams are thrown out, regardless of whether they were to be retained in or excluded from the model. An initial exploration of the one list approach shows that the perplexity of these models (for this data) is essentially the same as the weighted difference perplexities of models where approximately equal numbers of bigrams and trigrams were retained. Dividing the weighted difference factor for the bigrams by two (since they take up twice the space of a trigram in our current implementation), hence increasing their likelihood to be excluded from the model, shows no effect on model perplexity, though significantly more trigrams are retained in the model. Also, hand-picking the ratio of bigrams to trigrams to be 1:5 and 5:1 (rather than 1:1) results in worse perplexity measures than the 1:1 ratio. Further studies are required to determine if there is an optimal ratio of bigrams and trigrams that minimize model perplexity.

As a second topic of further investigation, an additional factor may prove helpful in the weighted difference measure. If a time coefficient is somehow added into the equation, then the effects of time shift in the training text may be accounted for. More recent text could be weighted more heavily, meaning that those trigrams and bigrams would have a greater chance of surviving the pruning step and be retained in the model. In this way, newer vocabulary and more current topics of interest could be reflected in the model, as they will be more likely to occur during the model's use. However, similar attempts to factor data time shift into vocabulary selection failed to show a significant effect [3].

References

[1] F. Jelinek, "Self Organized Language Modeling for Speech Recognition", in *Readings in Speech Recognition*, Alex Waibel and Kai-Fu Lee (Eds.), Morgan Kaufmann, 1989.

[2] S.M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer", in *IEEE Transactions on Acoustics, Speech and Signal Processing*, volume ASSP-35, pages 400-401, March 1987.

[3] R. Rosenfeld, "Optimizing Lexical and N-gram Coverage Via Judicious Use of Linguistic Data", Eurospeech 95, pp. 1763–1766. See file://localhost/afs/cs.cmu.edu/user/roni/WWW/vocov-eurospeech95-proc.ps.

[4] R. Rosenfeld, "The CMU Statistical Language Modeling Toolkit, and its use in the 1994 ARPA CSR Evaluation", in *Proc. ARPA Spoken Language Technology Workshop*, Austin, TX, January 1995. See http://www.speech.cs.cmu.edu/speech/SLM_info.html.