LANGUAGE MODEL SIZE REDUCTION BY PRUNING AND CLUSTERING

Joshua Goodman
Speech Technology Group
Microsoft Research
Redmond, Washington 98052, USA
joshuago@microsoft.com
http://research.microsoft.com/~joshuago

Jianfeng Gao
Natural Language Group
Microsoft Research China
Beijing 100080, P.R.C
jfgao@microsoft.com
http://www.microsoft.com/china/research

ABSTRACT

Several techniques are known for reducing the size of language models, including count cutoffs [1], Weighted Difference pruning [2], Stolcke pruning [3], and clustering [4]. We compare all of these techniques and show some surprising results. For instance, at low pruning thresholds, Weighted Difference and Stolcke pruning underperform count cutoffs. We then show novel clustering techniques that can be combined with Stolcke pruning to produce the smallest models at a given perplexity. The resulting models can be a factor of three or more smaller than models pruned with Stolcke pruning, at the same perplexity. The technique creates clustered models that are often *larger* than the unclustered models, but which can be pruned to models that are *smaller* than unclustered models with the same perplexity.

1. INTRODUCTION

Language models for large vocabulary speech recognizers and other applications are typically trained on hundreds of millions or billions of words. An uncompressed language model is typically comparable in size to the data on which it is trained. Some form of size reduction is therefore critical for any practical application. Many different approaches have been suggested for reducing the size of language models, including count-cutoffs [1], Weighted Difference pruning [2], Stolcke pruning [3], and clustering [4]. In this paper, we first present a comparison of these various techniques, and then we demonstrate a new technique that combines a novel form of clustering with Stolcke pruning, performing up to a factor of 3, or more, better than Stolcke pruning alone.

None of the techniques we consider are loss-less. Therefore, whenever we compare techniques, we do so by comparing the size reduction of the techniques at the same perplexity. We begin by comparing count-cutoffs, Weighted Difference pruning, Stolcke pruning, and variations on IBM pruning. All of our experiments are performed on both an English corpus – Wall Street Journal – and a Chinese newswire database. To our knowledge, no direct comparison of clustering versus count cutoffs or any of the other techniques has previously been done – we show that count cutoffs in isolation are much more effective than clustering in isolation. We also show that Weighted Difference and Stolcke pruning actually, in a few cases, underperform count cutoffs when only a small amount of pruning is being done.

Next, we consider combining techniques, specifically Stolcke pruning and a novel clustering technique. The clustering technique is surprising in that it often first makes the model *larger* than the original word model. It then uses Stolcke pruning to prune the model to one that is *smaller* than a standard Stolcke-pruned word model of the same perplexity.

2. PREVIOUS WORK

There are four well-known previous techniques for reducing the size of language models. These are count-cutoffs, Weighted Difference pruning, Stolcke pruning, and IBM clustering.

The best known and most commonly used technique is count-cutoffs. When creating a language model estimate for a probability of a word *z* given the two preceding words *x* and *y*, typically a formula of the following form is used:

$$P(z \mid xy) = \begin{cases} \dfrac{C(xyz) - D(C(xyz))}{C(xy)} & \text{if } C(xyz) > 0 \\ \alpha(xy) P(z \mid y) & \text{otherwise} \end{cases}$$

The notation *C(xyz)* indicates the count of *xyz* – the number of occurrences of the word sequence *xyz* in the training data. The function α is a normalization constant. The function *D(C(xyz))* is a discount function. It can, for instance, have constant value, in which case the technique is called "Absolute Discounting" or it can be a function estimated using the Good-Turing method, in which case the technique is called Good-Turing or Katz smoothing.

In the count cutoff technique, a cutoff, say 3, is picked, and all counts $C(xyz) \leq 3$ are discarded. This can result in significantly smaller models, with a relatively small increase in perplexity.

In the Weighted Difference method, the difference between trigram and bigram, or bigram and unigram probabilities is considered. For instance, consider the probability *P(City|New York)* versus the probability *P(City|York)* – the two probabilities will be almost the same. Thus, there is very little to be lost by pruning *P(City|New York)*. On the other hand, in a corpus like the Wall Street Journal, *C(New York City)* will be very large, so the count would not typically be pruned. The Weighted Difference method can therefore provide a significant advantage. In particular, the weighted difference method uses the value

$$[C(xyz) - D(C(xyz))] \times [\log P(z \mid xy) - \log P(z \mid y)]$$

For simplicity, we give the trigram equation here; an analogous equation can be used for bigrams, or other *n*-grams. Some pruning threshold is picked, and all trigrams and bigrams with a value less than this threshold are pruned. Seymore and Rosenfeld [2] did an extensive comparison of this technique to

count cutoffs, and showed that it could result in significantly smaller models than count cutoffs, at the same perplexity.

Stolcke pruning can be seen as a more mathematically rigorous variation on this technique. In particular, our goal in pruning is to make as small a model as possible, while keeping the model as unchanged as possible. The Weighted Difference method is a good approximation to this goal, but we can actually solve this problem exactly, using a relative entropy-based pruning technique, Stolcke pruning. The increase in relative entropy from pruning is

$$-\sum_{x,y,z} p(xyz)[\log P'(z \mid xy) - P(z \mid xy)]$$

Here, *P'* denotes the model after pruning, *P* denotes the model before pruning, and the summation is over all triples of words *x,y,z*. Stolcke shows how to efficiently compute the contribution of any particular trigram $P(z|xy)$ to the expected increase in entropy. A pruning threshold can be set, and all trigrams or bigrams that would increase the relative entropy less than this threshold are pruned away. Stolcke shows that this approach works slightly better than the weighted difference method, although in most cases, the two models end up selecting the same *n*-grams for pruning.

The last technique for compressing language models is clustering. In particular, IBM [4] showed that a clustered language model could significantly reduce the size of a language model with only a slight increase in perplexity. Let z^l represent the cluster of word *z*. The model used was of the form $P(z^l|x^ly^l) \times P(z|z^l)$. To our knowledge, no comparison of clustering to any of the other three techniques has been done.

3. PRUNING AND CLUSTERING COMBINED

Our technique is essentially a generalization of IBM's clustering technique, combined with Stolcke pruning. However, the actual clustering we use is somewhat different than might be expected. In particular, in many cases, the clustering we use first *increases* the size of the model. It is only after pruning that the model is smaller than a pruned word-based model of the same perplexity.

The clustering technique we use creates a binary branching tree with words at the leaves. By cutting the tree at a certain level, it is possible to achieve a wide variety of different numbers of clusters. For instance, if the tree is cut after the 8^{th} level, there will be roughly 2^8=256 clusters. Since the tree is not balanced, the actual number of clusters may be somewhat smaller. We write z^l to represent the cluster of a word *z* using a tree cut at level *l*. Each word occurs in a single leaf, so this is a hard clustering system, meaning that each word belongs to only one cluster. We actually use two different clustering trees, one for the *"z"* position, and one optimized for the *"y"* position (which is also used for the *"x"* position.) [5]

We notice that there is no need for the sizes of the clusters used in different positions to be the same. In particular, we use a model of the form $P(z^l|x^iy^j) \times P(z|x^ky^kz^l)$. We call this model the "Both Clusters" technique. We also create a model of the form $P(z^l|xy) \times P(z|xyz^l)$. We call this the "Predict Clusters" technique.

Optimizing such a large number of parameters is potentially overwhelming. In particular, consider a model of the type $P(z^l|x^iy^j) \times P(z|x^ky^kz^l)$. There are 5 different parameters that need to be simultaneously optimized for a model of this type: *j, k, l*, the pruning threshold for $P(z^l|x^iy^j)$, and the pruning threshold for $P(z|x^ky^kz^l)$. Rather than try a large number of combinations of all 5 parameters, we give an alternative technique that is significantly more efficient. Simple math shows that the perplexity of the overall model $P(z^l|x^iy^j) \times P(z|x^ky^kz^l)$ is equal to the perplexity of the cluster model $P(z^l|x^iy^j)$ times the perplexity of the word model $P(z|x^ky^kz^l)$. The size of the overall model is clearly the sum of the sizes of the two models. Thus, we try a large number of values of *j, l*, and a pruning threshold for $P(z^l|x^iy^j)$, computing sizes and perplexities of each, and a similarly large number of values of *l, k*, and a separate threshold for $P(z|x^ky^kz^l)$. We can then look at all compatible pairs of these models (those with the same value of *l*) and quickly compute the perplexity and size of the overall models. This allows us to relatively quickly search through what would otherwise be an overwhelmingly large search space.

4. EXPERIMENTAL RESULTS AND DISCUSSION

We performed our experiments on two different corpora. The first was a subset of the Wall Street Journal corpus. In particular, we used the first ten million words of the Wall Street Journal corpus for training data. For heldout data, we used every 50th sentence, taken from a set of 250,000 words from the end of the corpus. For test data, we used every 50th sentence taken from a disjoint set of 1,000,000 words at the end of the corpus.

For most of our experiments, we built a large number of models. Rather than graph all points of all models together, we show only the outer envelope of the points. That is, if for a given model type and a given point there is some other point of the same type with both lower perplexity and smaller size than the first point, then we do not graph the first, worse point.

We built a very large number of models for English as follows. First, for experiment with clusters only, we tried most models of the form $P(z^l|x^iy^j) \times P(z|x^ky^kz^l)$ for values $2 \le j,k,l \le 12$. For cutoffs, we tried 167 different combinations of cutoffs, where the trigram cutoff varied between 0 and 1024, in increments of about 50% and the bigram cutoff was various fractions of the trigram cutoff, between 0.1 and 1.2 times the trigram cutoff. For weighted difference pruning and Stolcke pruning, we tried a large range of parameters sufficient to cover the same perplexities as covered by the count cutoffs. The size was measured as the total number of parameters of the system: one parameter for each bigram and trigram that was not thresholded, as well as one parameter for each normalization parameter α that was needed, and one parameter for each unigram. In the pruning experiments, bigrams and trigrams were both pruned, unigrams never were. This resulted in the smallest possible number of parameters being equal to the vocabulary size, approximately 60,000 words.

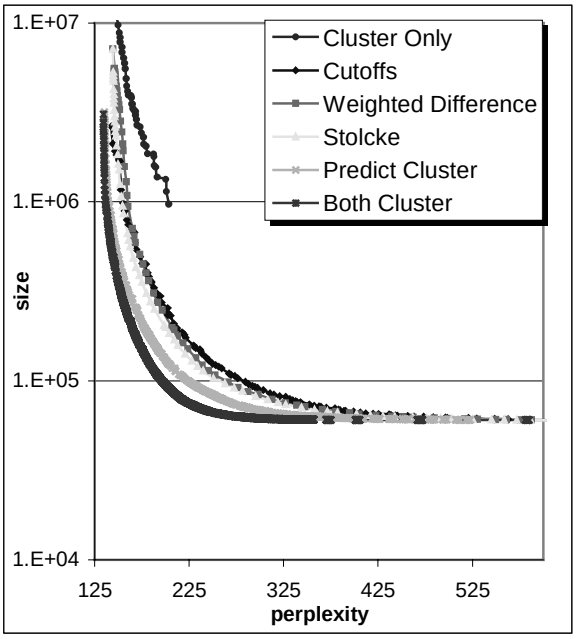


Figure 1: Perplexity versus size on WSJ data

Figure 1 shows the results of these experiments. Unfortunately, because of the very large range of sizes, it is difficult to resolve detail. The main result that can be observed is that the cluster-only technique is by far the worst. The cluster-only technique is roughly an order of magnitude worse than the others. In order to show more detail, we also plotted relative sizes. Each technique was plotted compared to the Both Cluster technique.

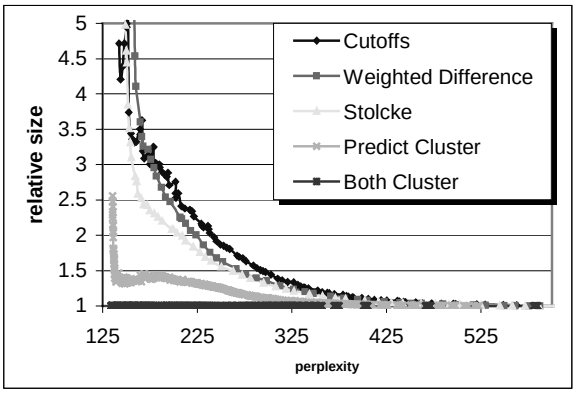


Figure 2: Perplexity versus relative size on WSJ data

The main result to notice here is that over a wide range of values, the Both Cluster technique produces models that are at most 2/3 the size of Stolcke-pruned models with the same perplexity. In many cases, the models are half the size or less. This is a much larger improvement, than, say, the relatively small improvement of Stolcke pruning over count cutoffs alone.

The other interesting result is the surprisingly good performance of count cutoffs, better than previously reported, and, at low thresholding values, marginally better than Weighted Difference pruning, and overlapping with Stolcke pruning. We have a few explanations for this. First, Stolcke did not compare his technique directly to count cutoffs, but only to Weighted Difference pruning. A close look at the paper comparing Weighted Difference pruning [2] to count cutoffs shows that very few points were compared at the low-pruning end of the chart. Also, in the previous work, rather trying a large number of reasonable values for bigram and trigram cutoffs, and then looking at the best ones, bigrams and trigrams were pruned in such a way as to have the same number of non-zero parameters.

The most interesting analysis is to look at some sample settings of the parameters of the Both Clusters system, as shown in table 1. The value "all" for k means that the tree was cut at infinite depth, i.e. each cluster contained a single word. The "prune" column indicates the Stolcke pruning parameter used.

l	j	prune $P(z^l\|x^jy^j)$	k	prune $(z\|x^ky^kz^l)$	perplex	size
7	7	2560	13	3072	292.7	62077
7	9	896	12	768	249.0	66938
7	9	128	12	160	204.7	90382
8	10	112	all	128	194.3	102436
8	12	56	16	64	173.9	153799
8	12	28	all	42	164.1	203372
7	12	14	16	20	153.3	300193
7	13	10	all	10	146.0	452421
7	15	3	all	4	136.3	1007055
7	16	1.5	all	1	134.2	2533680

Table 1: Sample parameter settings for $P(z^l|x^jy^j) \times P(z|x^ky^kz^l)$

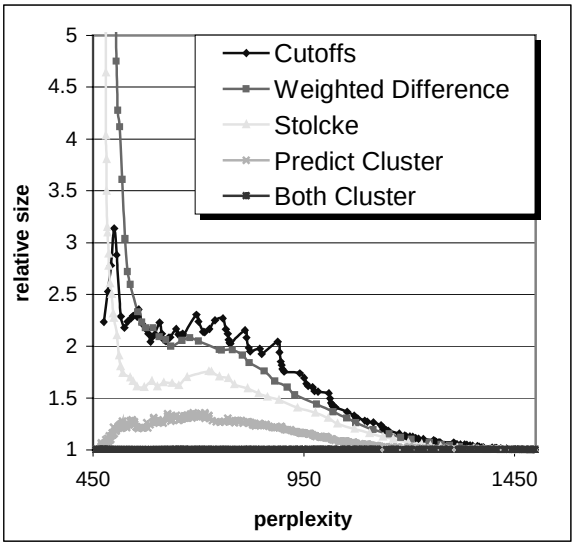
First, notice that the two pruning parameters (in columns 3 and 5) tend to be very similar. This is good, since applying the theory of relative entropy pruning predicts that the two pruning parameters should actually have the same value.

Next, compare our model, of the form $P(z^l|x^jy^j) \times P(z|x^ky^kz^l)$ to traditional IBM clustering, of the form $P(z^l|x^jy^j) \times P(z|z^l)$, which is equal to $P(z^l|x^jy^j) \times P(z|x^0y^0z^l)$. Traditional IBM clustering makes two assumptions that we see are suboptimal. First, it assumes that $j=l$. We see that the best results come from unequal settings of j and l. Second, more importantly, IBM clustering assumes that $k=0$. We see that not only is the optimal setting for k not 0, it is typically the exact opposite: *all* (in which case $P(z|x^ky^kz^l) = P(z|xyz^l)$), or 16, which is very similar. That is, we see that words depend on the previous words, and that an independence assumption is a poor one. Of course, many of these word dependencies are pruned away – but when a word does depend on something, the previous words are better predictors than the previous clusters. The other important finding here is that for most of these settings, the unpruned model is actually larger than a normal trigram model – whenever k=all or 16, the unpruned model $P(z^l|x^jy^j) \times P(z|x^ky^kz^l)$ is actually larger than an unpruned model $P(z|xy)$.

This analysis of the data is very interesting – it implies that the gains from clustering are not from compression, but rather from capturing structure. Factoring the model into one in which the

cluster is predicted first, and then the word is predicted given the cluster allows the structure and regularities of the model to be found. This larger, better structured model can be pruned more effectively.

We also performed experiments on a Chinese Newswire corpus of about 12 million characters, with a 65,502 word vocabulary.



It is interesting to compare the two sets of results. The main conclusion is that for the two corpora, in two different languages, the results are qualitatively very similar. In particular, again for Chinese, Stolcke pruning is better than Weighted Difference pruning; at very low threshold values, count cutoffs outperform both. The Both Cluster technique works significantly better than Stolcke pruning alone, producing models that are at least a third smaller than Stolcke pruning alone at the same perplexity over a range of perplexity values. The most noticeable difference is that the Predict Cluster technique and the Both Cluster technique converge here, rather than diverging sharply, as in English. This should not be interpreted too strongly. Recall that Predict Cluster is a special case of Both Cluster. In the Chinese data, for the largest models, it turned out to be optimal to use j=k=all, i.e. no clustering. For the English data, marginally better performance at a given size could be achieved with j=16, k=all. This marginal difference in the asymptotic best perplexity leads to large relative differences in sizes. Indeed, for the same reason, the other steep divergences at the left sides of the graphs should not be over-emphasized either.

Why does our system work as well as it does? Consider an example: *P(Tuesday | party on)*. We might prune this model to one such as *P(WEEKDAY|EVENT PREPOSITION)* $\times$*P(Tuesday|WEEKDAY)*. Clearly, in general, it is not necessary to keep separate probabilities for *P(Wednesday | party on WEEKDAY)* and *P(Thursday | party on WEEKDAY)*. On the other hand, consider *P(Friday | God it's)* as in the phrase "Thank God it's Friday". Clearly, *P(Friday | God it's WEEKDAY)* differs markedly from *P(Monday | God it's WEEKDAY)*. This ability to capture both idiomatic usage in the word model, and standard patterns in the cluster model, appears important. An examination of unpruned WSJ *n*-grams is consistent with this hypothesis, though the unpruned *n*-grams tend to be more like *P(commission | and exchange)*, as in the phrase "Securities and exchange commission."

5. CONCLUSIONS

We have performed a fairly thorough comparison of different types of language model size reduction. One area we have not explored in this paper is the actual representation of the language model, which is generally more of an engineering issue. However, there are interesting interactions between the language model compression technique and the language model representation. For instance, with count cutoff techniques, one can easily use the bigram data structure both to store *P(z|y)* and α*(yz)* in a single structure. Similarly, there are interactions between language model representation, and some implementions of tree decoders for speech recognition. These interactions cannot be ignored in practical systems.

We have shown several interesting results. They include the fact that when count cutoffs are well optimized, they are better in some cases than previously reported. We have done the first systematic comparison of clustering techniques to pruning techniques, and shown that the pruning techniques outperform the clustering techniques, when used separately. Finally, we showed that by combining clustering and Stolcke pruning techniques in a novel way, we can achieve significantly better results than using Stolcke pruning alone. Our clustering technique is particularly interesting because it typically *increases* the size of the unpruned model, while resulting in an overall *decrease* in pruned model size. Similarly, it is interesting because it shows that using many different cluster sizes for different purposes is helpful, and that the cluster sizes are different than the conventional wisdom might suggest.

6. REFERENCES

1. F. Jelinek, "Self Organized Language modeling for Speech Recognition", in *Readings in Speech Recognition*, A. Waibel and K. F. Lee(Eds.), Morgan Kaufmann, 1990
2. K. Seymore, R. Rosenfeld. "Scalable backoff language models", *Proc. ICSLP*, Vol. 1., pp.232-235, Philadelphia, 1996
3. A. Stolcke, "Entropy-based Pruning of Backoff Language Models" *Proc. DARPA News Transcription and Understanding Workshop*, 1998, pp. 270-274, Lansdowne, VA.
4. P. F. Brown,V. J. DellaPietra, P. V. deSouza, J. C., Lai, R. L. Mercer. "Class-based n-gram models of natural language". *Computational Linguistics* 1990 (18), 467-479.
5. H. Yamamoto, Y. Sagisaka, "Multi-class Composite N-gram based on Connection Direction", *Proc. ICASSP*, May, 1999, Phoenix, Arizona.