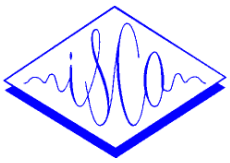

6th International Conference on Spoken Language Processing (ICSLP 2000)
Beijing, China
October 16-20, 2000

IMPACT OF BUCKETING ON PERFORMANCE OF LINEARLY INTERPOLATED LANGUAGE MODELS

K. Visweswariah, H. Printz, M. Picheny

IBM
Thomas J. Watson Research Center
Yorktown Heights, NY 10598
e-mail: {kv1,printz,picheny}@us.ibm.com

ABSTRACT

N-gram models are used to model language in various applications. For large vocabularies, even a very large corpus is insufficient to estimate a raw ratio-of-counts trigram model. One common way to overcome this problem is by linear interpolation of the trigram model with lower order models. The interpolation weights can be varied as a function of the current history, to reflect the confidence we have in the estimates of various orders. Since the number of histories is large we cannot hope to estimate a set of weights for each history. Thus sets of histories are tied together and the same weights are used for all histories within the set. In this paper we study the effect of the algorithm used to tie together the various histories. We report word error rate (WER) results on a large-vocabulary speech recognition task.

1. INTRODUCTION

Consider a trigram language model where the probability of a word is estimated based on the the last two words. The number of parameters to be estimated in this case is roughly $|V|^3$ where V is the vocabulary. For large vocabularies the number of parameters is too high to form reliable estimates, even from a very large corpus of data. To deal with this problem we smooth this model with those of lower order.

Various ways of performing this smoothing are known; for example Katz smoothing [1], Witten-Bell smoothing [2] and Kneser-Ney smoothing [3]. For a more exhaustive list of smoothing techniques, and comparison of the techniques see [4]. One way of doing this [5] is by linear interpolation as given by

$$P(w_i|w_{i-2},w_{i-1}) = \lambda_0 f_0(w_i) + \lambda_1 f_1(w_i) + \lambda_2 f_2(w_i|w_{i-1}) + \lambda_3 f(w_i|w_{i-2},w_{i-1}).$$

In the above equation $f_0(w_i) = 1/|V|$ and

$$f_j(w_i|w_{i-j+1}^{i-1}) = \frac{C(w_{i-j+1}^{i})}{\sum_{v\in V} C(w_{i-j+1}^{i-1}v)},$$

where $C(w_{i-j}^{i})$ denotes the number of times the word sequence w_{i-j}^{i} occurs in a corpus $\mathcal{C}$. If we require that $\sum \lambda_i = 1$, then $P(\cdot|w_{i-2},w_{i-1})$ will be a distribution for each (w_{i-2},w_{i-1}).

We want to choose the λs to reflect our confidence in the estimates of various orders. Intuitively, the greater the number of times a particular context occurs the higher the reliability of the estimates that we have for the distribution in that context. We would thus expect the values of the λs to depend on $C(w_{i-1})$ and $C(w_{i-2},w_{i-1})$. The number of distinct pairs $C(w_{i-1}), C(w_{i-2},w_{i-1})$ is so high that estimating λs for each pair of counts is impossible. Thus, the space of pairs of integers is partitioned into a fixed number (say k) of buckets, and the λs are estimated in each bucket independently. If we use the same corpus $\mathcal{C}$ (that was used to obtain the counts) to estimate the λs then we will have λ_3 being set to one and the other weights going to zero. Thus, within each bucket the λs are chosen to maximize likelihood of a held out set $\mathcal{H}$ different from the corpus $\mathcal{C}$ from which the counts were obtained. In this paper we will investigate various schemes of forming these buckets and study the effect on the quality of the language model from the point of view of its performance in a speech recognizer.

2. DESCRIPTION OF METHODS

In this section we will describe the various algorithms for forming buckets that we studied. There are two factors which have to be taken into account in the formation of the buckets

- there is enough data in the held-out set $\mathcal{H}$ for each bucket to be trained
- the buckets are smaller in size where the λs change rapidly as a function of $C(w_{i-1})$ and $C(w_{i-2},w_{i-1})$.

2.1. Method 1

This method tries to place an equal amount of data in each bucket, and does so by first forming a partition based on the count of the last bigram and then further dividing these sets only if a particular set contains a large number of data points.

Let us assume that we want to form k buckets. Let $N(j)$ be the number of bigrams (u, v) in $\mathcal{H}$ such that $C(u, v) = j$. Also let $N^{\text{cum}}(j)$ be the number of bigrams (u, v) in $\mathcal{H}$ such that $C(u, v) \leq j$ and N the total number of bigrams in $\mathcal{H}$. We form the buckets by first partitioning the space according to the count of the last bigram. Thus we find integers $i_1, \ldots, i_v$ such that

$$\sum_{j=i_l}^{i_{l+1}} N(j) \simeq \frac{N - N^{\text{cum}}(i_l)}{k - l + 1}.$$

Thus we divide the space according to the count of the last bigram such that each region contains approximately $1/k$ of total data. If any strip contains more than a given threshold T of events, then the strip is further divided based on $c(w_{i-1})$, in such a way that within that strip the data is split equally. In practice this occurs only for the case $c(w_{i-2}, w_{i-1}) = 0$. For a held out corpus of a million words, a typical value of T is 10^4 when k the number of buckets is 150. Figure 1 is a schematic representation of buckets obtained using this method.

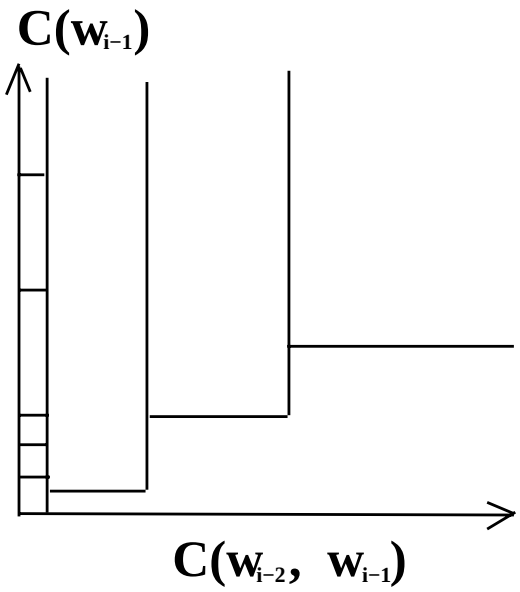

Figure 1: Typical buckets obtained using Method 1. The vertical strips represent the initial division created by the algorithm. Further subdivision may then take place within each strip.

2.2. Method 2

We again try to place an equal amount of data in each bucket, but this time we first form a partition of the space of size $\sqrt{k}$ based on the count of the last bigram and then further partition each of these sets into $\sqrt{k}$ sets based on the count of the last word. At each stage we do so trying to ensure that each bucket has approximately N/k of the data points. Figure 2 is a representation of buckets obtained using this scheme, with $k = 9$.

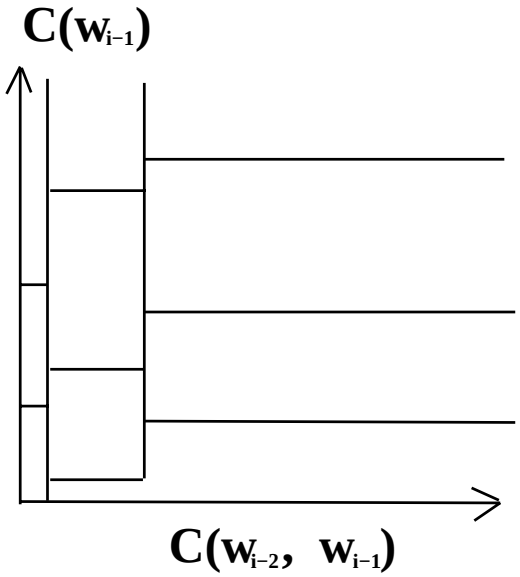

Figure 2: Typical buckets obtained using Method 2 ($k = 9$). There are 3 vertical strips and a further division into 3 regions within each strip.

2.3. Method 3

The previous two methods did not consider how quickly the λs change as a function of $C(w_{i-1})$ and $C(w_{i-2}, w_{i-1})$. Ideally we would like more buckets in those regions where the λs change rapidly, so that we can track that variation. We used the empirically observed variation of the λs in designing the buckets. We try to design the buckets so that the variation of λs over a bucket is approximately the same for all buckets. This will ensure that where λs change quickly the buckets are smaller. Of course we need to be able to estimate the parameters within each

bucket and so we need to counterbalance this by ensuring that we have enough data in the held out set in each bucket.

2.4. Method 4

Here we assume that the confidence we have in the second order model depends only on the count of the last bigram. We then assign the remaining weight to the lower order models, which model gets what fraction of the remaining weight depending only the count of the last word now. This method is essentially the recursive method described in [5].

We form the buckets based on the assumption that λ_3 depends only on the count of the last bigram. We also assume that for $j = 0, 1, 2$, the dependence of λ_j is of the form

$$\lambda_j(C(w_{i-1}), C(w_{i-2}, w_{i-1})) = g_j(C(w_{i-1})) \times (1 - \lambda_3(C(w_{i-2}, w_{i-1}))).$$

This implies that we can increase the number of buckets without increasing the number of parameters that we have to train because now the λ values in various buckets are not independent. Thus essentially we can have k^2 buckets with the same number of parameters as we had before with k buckets. We now estimate the buckets in each dimension independently, we first from buckets and estimate the lambda values for each bucket for the uniform, unigram and bigram model. Once we have this we form the buckets for $C(w_{i-2}, w_{i-1})$ and estimate λ_3. In each dimension we form the k buckets so that we have approximately equal amounts of data in each bucket.

Figure 3 is a representation of buckets obtained using this scheme, the number of parameters is well below the number of regions the space is divided into.

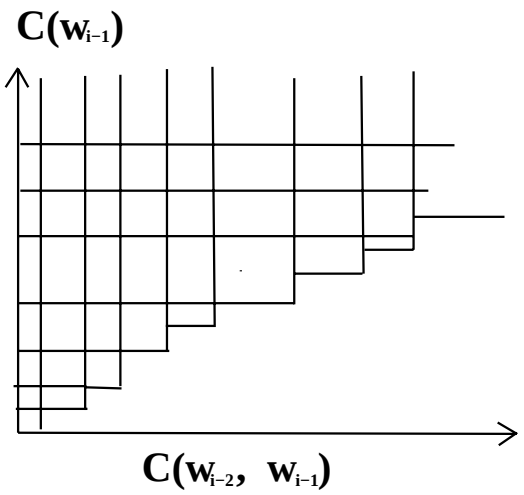

Figure 3: Typical buckets obtained using Method 4. Even though we have 9×9 regions the number of parameter is as if we had 9 regions.

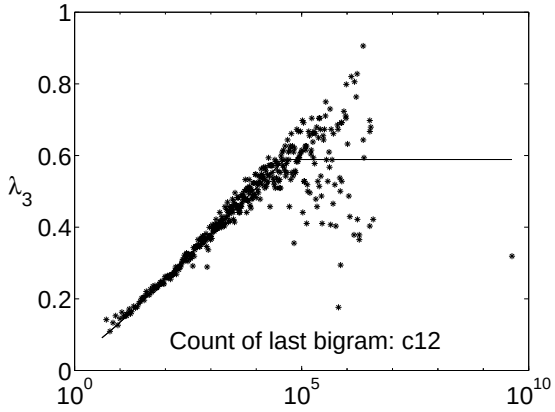

Figure 4: Fit of $g_3(x)$ for λ_3

2.5. Method 5

In this method we dispense with buckets, and try to fit a parametric form to the λ values. We make the same assumptions as in the method above about the functional forms of the λs:

$$\lambda_3(C(w_{i-1}), C(w_{i-2}, w_{i-1})) = g_3(C(w_{i-2}, w_{i-1}))$$

and

$$\lambda_j(C(w_{i-1}), C(w_{i-2}, w_{i-1})) = g_j(C(w_{i-1})) \times (1 - \lambda_3(C(w_{i-2}, w_{i-1})))$$

for $j = 0, 1, 2$. We find parametric forms (and parameter values) for g_j which fit empirical data, and then use these functions instead of the buckets. The parametric forms that we found to be a reasonable fit for the data, by inspection were:

$$g_3(x) = \min(a_3 \log x + b_3, c_3)$$

for $x \geq 1$ (clearly $g_3(0) = 0$),

$$g_2(x) = 1 - \frac{a_2}{(b_2 + x)^{c_2}}$$

and

$$g_j(x) = \frac{a_j}{(b_j + x)^{c_j}}$$

for $j = 0, 1$. Thus for each weight we have three parameters to estimate. The parametric forms that we obtain for λ_3 and for λ_2 are shown in Figure 4 and Figure 5 respectively. Figure 5 shows the variation of λ_2 when $C(w_{i-2}, w_{i-1}) = 0$. The data plotted are the lambda values for the buckets in that region.

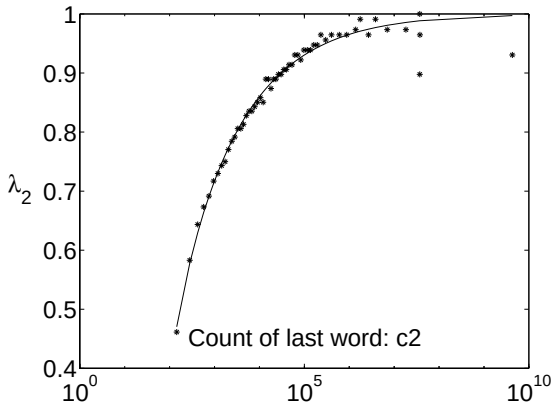

Figure 5: Fit of $g_2(x)$ for λ_2

3. EXPERIMENTAL SETUP AND RESULTS

We conducted experiments with the bucketing scheme described above with a language model built from a training corpus of about 10^9 words. The size of the held out corpus was about 5×10^6 words. Once the buckets were formed the lambda values in each bucket were obtained using the EM algorithm. In each case the language model obtained is used to decode a test set consisting of about 40k words. All parameters except the buckets and the weights for the various order language models are kept constant over the experiments conducted. The task is a large-vocabulary, read-speech recognition task.

The results obtained when we used 150 buckets (except for Method 5 which does not use any buckets) are given in Table 1: We see that there is not much

Bucketing Method	Word Error Rate
Method 1	9.19
Method 2	9.27
Method 3	9.28
Method 4	9.19
Method 5	9.17

Table 1: Comparison of word error rates for various bucketing schemes

variation in the error rate as a function of the method used to form the buckets. We found that we can use a scheme with no buckets to obtain the essentially the same performance as scheme with buckets. We also note that of the bucketing schemes the one that performs the best (Method 1) is the one that divides the space more finely based on $C(w_{i-2}, w_{i-1})$ rather than $C(w_{i-1})$.

We also varied the number of buckets constructed using Method 1 and the results are given in Table 2.

Number of buckets	Word Error Rate
1	9.46
150	9.19
450	9.05
1000	9.29

Table 2: Comparison of word error rates for Method 1, for different numbers of buckets

4. CONCLUSION

We see that the shape of the buckets used in weighting estimates of different orders does not significantly affect the error rates. It also seems much more important to form the buckets with a fine division based on the count of the last bigram; divisions based on the count of the last word seem to play a much smaller role. The number of buckets used has a small effect on the error rate: there is a variation of about 4% relative from the best case to worst case. The number of parameters used can be reduced by using a parametric model for the λs without affecting the error rates.

5. REFERENCES

[1] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoust. Speech Sig. Proc.*, vol. 35, no. 3, pp. 400–401, March 1987.

[2] I. H. Witten and T. C. Bell, "The zero-frequency problem: Estimating the probability of novel events in adaptive text compression," *IEEE Trans. Inform. Theory*, vol. 37, no. 4, pp. 1085–1094, July 1991.

[3] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *IEEE Int. Conf. Acoust. Speech Sig. Proc.*, volume 1, pp. 181–184, 1995.

[4] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.

[5] F. Jelinek, *Statistical methods for speech recognition*, The MIT Press, Cambridge MA, 1997.