EFFICIENT CONSTRUCTION OF LONG-RANGE LANGUAGE MODELS USING LOG-LINEAR INTERPOLATION

E.W.D.Whittaker[*] and *D.Klakow*

Philips Research Laboratories, Weißhausstr. 2
D-52066 Aachen, Germany
{edward.whittaker,dietrich.klakow}@philips.com

ABSTRACT

In this paper we examine the construction of long-range language models using log-linear interpolation and how this can be achieved effectively. Particular attention is paid to the efficient computation of the normalisation in the models. Using the Penn Treebank for experiments we argue that the perplexity performance demonstrated recently in the literature using grammar-based approaches can actually be achieved with an appropriately smoothed 4-gram language model. Using such a model as the baseline, we demonstrate how further improvements can be obtained using log-linear interpolation to combine distance word and class models. We also examine the performance of similar model combinations for rescoring word lattices on a medium-sized vocabulary Wall Street Journal task.

1. INTRODUCTION

Statistical language models (LMs) have typically been used in automatic speech recognition systems. In such an application they are used to provide an estimate of the likelihood of competing word sequences. They thus reduce the search space of sequences representing the spoken utterance, to those which are deemed statistically likely according to the language and acoustic models. More recently, language models have also found uses in information retrieval [1] and statistical translation [2] among others. In each application, however, the aim of building the best statistical representation of a language with a limited amount of training data is a common theme. It is also becoming more common to incorporate disparate and longer-ranging language dependencies into language models to complement and improve upon the limited range and expression of the word N-gram model. To this end, maximum entropy models [3] and log-linear interpolation [4] (LLI) of language models have been used with considerable success. The maximum entropy technique combines different features into a single model and constructs a probability distribution that satisfies a set of linear constraints. Log-linear interpolation combines separate language models each of which may model different language dependencies and indeed may have been trained on different data. In particular, the log-linear interpolation technique has been found to combine distance word and class bigrams significantly better than linear interpolation and almost as well as maximum entropy models when the same dependencies are used [5].

In this paper we are primarily interested in how to efficiently build a high-quality long-range LM for eventual use in various natural language tasks. We therefore focus on using perplexity as the evaluation metric and on how such models compare with grammar-based methods trained and evaluated on the same data. As a complement to the perplexity focus of the paper, we also present perplexity and word error rate results on a medium-sized vocabulary speech recognition task.

[*]E.W.D.Whittaker was supported in this research by an EU Marie Curie Industrial Research Fellowship.

2. LOG-LINEAR INTERPOLATION

Log-linear interpolation of language models was introduced in [4]. The method is related to maximum entropy modelling but retains the flexibility and same number of free parameters as the frequently used method of linear interpolation. The general form of model combination is as follows:

$$P(w_0 \mid h) = \frac{1}{Z(h)} \prod_i P_i(w_0 \mid h)^{\lambda_i} \qquad (1)$$

in which there is no explicit constraint on the exponents λ_i. This model can also be interpreted as a linear interpolation of log probabilities with an additional normalisation term [4].

In this paper we examine the construction of a log-linear interpolation of a baseline 4-gram LM, several distance bigram LMs and a unigram LM:

$$P(w_0 \mid h) = \frac{1}{Z(h)} P^{\lambda_{4g}}_{\text{4-gram}}(w_0 \mid w_{-3}^{-1}) \prod_{i=3}^{10} \left\{ \frac{P_i(w_0 \mid w_{-(i+1)})}{P(w_0)} \right\}^{\lambda_i} \qquad (2)$$

The motivation for including the unigram LM is as follows: if the estimate for a distance bigram is not significantly different to the unigram distribution its inclusion has little effect. On the other hand, if there is a particularly strong correlation between two words at a particular distance and the estimate differs significantly from the unigram estimate, this effect will be relatively strong. This phenomenon is in stark contrast to linear interpolation which smoothes the effects of strong correlations as well as weak ones. Dependencies that have similar distributions to the unigram can therefore have a detrimental effect and the predictive power of the overall model can be weakened. In log-linear interpolation, on the other hand, including a unigram LM in the denominator cancels out the effect of the unigram information contained in the distance bigrams. A similar cancellation could be achieved with linear interpolation but with a corresponding difficulty in ensuring positive probabilities. In practice only one unigram model is used in the denominator and its exponent is normally constrained to be the sum of the exponents of the component distance bigram models.

The distance models which we use are themselves a linear interpolation of distance word bigram and distance class bigram LMs. This improves the robustness of the individual distance models but does not smooth the effect of correlations between words with a fixed separation.

3. EFFICIENT NORMALISATION

Computing the normalisation term $Z(h)$ is the most time consuming part of training the parameters of a log-linear model[1]. Each time one of the free parameters changes the normalisation must be re-computed. Since the optimisation procedure is performed on some held-out data, the normalisation terms must be re-computed for each unique history in the data. The computational complexity of the normalisation computation is therefore dependent on the size of the held-out data. Depending on the models that are being combined and the methods for performing the normalisation, the complexity may also be dependent on the size of the training data and the vocabulary size.

One method that can be used to reduce computation time is the caching of already computed normalisation terms for example in a tree structure built from occurring word contexts. This is typically more significant for computing the normalisation during lattice rescoring since there are many identical contexts for which the normalisation need only be computed once. Ultimately, the speed-ups will depend on the repetitiveness of the data that is used.

Below, we consider two variants for computing the normalisation efficiently. The first method assumes that the dependencies in the models that are being combined can be incorporated efficiently into separate tree structures. The second method is far more general and can be used to efficiently compute the normalisation when arbitrary models are combined, albeit not quite as efficiently as the first method.

3.1. Tree-structure method

The approach presented in this section is similar to the hierarchical training method for maximum entropy language models given in [6]. In this presentation we also exploit the structure of the component back-off language models in terms of back-off weights and the tree organisation of N-grams.

If we consider the case of log-linearly interpolating two different language models $P_A(w \mid h)$ and $P_B(w \mid h)$ with exponents λ_A and λ_B respectively, the normalisation for the null history at the unigram level is simply:

$$Z(\emptyset) = \sum_w P_A(w)^{\lambda_A} P_B(w)^{\lambda_B}. \tag{3}$$

The normalisation for a single word history w_{-1} given by

$$Z(w_{-1}) = \sum_w P_A(w \mid w_{-1})^{\lambda_A} P_B(w \mid w_{-1})^{\lambda_B}, \tag{4}$$

[1]The normalisation terms must also be computed during evaluation (e.g. recognition) where its efficiency is also likely to be an issue.

can be computed efficiently by propagating the normalisation value for the null history computed in Equation (3) and multiplying it by the back-off weights $\alpha_A(w_{-1})$ and $\alpha_B(w_{-1})$ for history w_{-1} in each of the component models. The appropriate updates to the normalisation for the explicit words that appear in the context w_{-1} in each language model must then be made, as follows:

$$\begin{aligned}
Z(w_{-1}) &= \alpha_A(w_{-1})^{\lambda_A} \alpha_B(w_{-1})^{\lambda_B} Z(\emptyset) + \\
&\sum_{\substack{w:N_A(w_{-1},w)>0 \\ w:N_B(w_{-1},w)>0}} \Big\{ P_A(w \mid w_{-1})^{\lambda_A} P_B(w \mid w_{-1})^{\lambda_B} \\
&- \alpha_A(w_{-1})^{\lambda_A} \alpha_B(w_{-1})^{\lambda_B} P_A(w)^{\lambda_A} P_B(w)^{\lambda_B} \Big\} + \\
&\sum_{\substack{w:N_A(w_{-1},w)>0 \\ w:N_B(w_{-1},w)=0}} \Big\{ P_A(w \mid w_{-1})^{\lambda_A} \alpha_B(w_{-1})^{\lambda_B} P_B(w)^{\lambda_A} \\
&- \alpha_A(w_{-1})^{\lambda_A} \alpha_B(w_{-1})^{\lambda_B} P_A(w)^{\lambda_A} P_B(w)^{\lambda_B} \Big\} + \\
&\sum_{\substack{w:N_A(w_{-1},w)=0 \\ w:N_B(w_{-1},w)>0}} \Big\{ \alpha_A(w_{-1})^{\lambda_A} P_A(w)^{\lambda_A} P_B(w \mid w_{-1})^{\lambda_B} \\
&- \alpha_A(w_{-1})^{\lambda_A} \alpha_B(w_{-1})^{\lambda_B} P_A(w)^{\lambda_A} P_B(w)^{\lambda_B} \Big\}.
\end{aligned} \tag{5}$$

The actual implementation also factors out the back-off weights from the summations to minimise unnecessary multiplication operations. The same development can straightforwardly be extended to longer histories and simply involves the recursive propagation of the normalisation from the level immediately above: for any given history node in the tree for which we wish to compute the normalisation, the normalisation from the history node one level higher in the tree is propagated down and multiplied by the back-off weights for the current history node from each component language model. Then, updates are performed according to the words in each language model that follow the given history that we are computing the normalisation for. Only a fraction of the total vocabulary will typically follow a given history in the language model and the longer the history the smaller this fraction will tend to be. This latter computation can be implemented efficiently by interleaving all the leaves from different language models corresponding to a particular history so that matching leaves in different models are handled simultaneously. This removes the need for any searching for the existence or absence of leaves in the component language models.

This method clearly relies on the average number of words following a given history to be significantly smaller than the total vocabulary size. In the degenerate case of almost all vocabulary words being present for a given history then at least twice as many computations are performed compared to simply computing the total sum of combined probabilities for a node. Unfortunately this degenerate case is inherently exhibited by distance word bigrams and also class models particularly when small numbers of classes are used.

3.2. Vector of probabilities method

The second method involves, for each component language model, the computation of a vector of probabilities for each word w in the vocabulary given a history h. The normalisation calculation simply involves computing the sum of products of each corresponding element in the component vectors. This is shown below for the case of two component language models:

$$Z(h) = \sum_{w} P_A(w \mid h)^{\lambda_A} P_B(w \mid h)^{\lambda_B}. \qquad (6)$$

This approach is more general than the method presented above since computing the probability vector efficiently can be treated independently for each component language model. The speed-up comes from caching the vectors of the exact or backed-off probabilities. The final sum of products computation for M component LMs $\left(\sum_w \prod_i^M P_i(w \mid h)\right)$ must be performed in every case but even for a large vocabulary this computation is extremely fast and linear in the vocabulary size and the number of models being combined.

4. EXPERIMENTS

The results of perplexity experiments on both Penn Treebank data [7] and a Wall Street Journal 5K task are presented in this section. The partitions of the Penn Treebank data, text processing and vocabulary definition are identical to those used in the experiments presented in [8, 9] to permit a fair comparison[2]. The 38M words of language modelling data for the Wall Street Journal 5K task is referred to as WSJ1 in this paper.

Language model combination is somewhat of a black art and testing which models can be combined to give an improvement and under what conditions requires a large number of variants to be evaluated. Previous experience advised using Kneser-Ney smoothing and optimising all discounting parameters of all component language models on the development data. For distance models beyond distance-5 a merged set of counts based on the distance-6 to distance-10 bigram statistics was used. This was motivated by issues of both data sparsity and the less constrained position dependencies between words separated by more than 5 other words. Another scenario was also considered whereby additional training data was used to augment the models evaluated on the Penn Treebank. This additional data, referred to here as WSJ2, comprised around 60M words and came from the Wall Street Journal portion of the North American Business News language model training data. This data occurred approximately four years after the Penn Treebank evaluation data. The WSJ2 data was only used to build distance models for distances greater than 5 by building two sets of all models, one on each set of data and linearly interpolating the component models. This was found to be significantly better than simply pooling the training data and building only one model.

Model construction for the experiments on the Penn Treebank data proceeded in the following incremental manner:

- Construct baseline 4-gram LM by linearly interpolating word 4-gram with 100-, 200- and 500-class 4-gram models using Kneser-Ney smoothing and optimised discount coefficients
- Construct distance word bigram LMs for distances $d = 1 \ldots 10$ with optimised discount coefficients. Merge bigram count trees for $d = 6 \ldots 10$
- Determine classifications of vocabulary into optimal number of classes for each distance $d = 1 \ldots 10$ by maximising leaving-one-out training set likelihood[3]
- Construct distance class bigrams for distances $d = 1 \ldots 10$
- Combine using linear interpolation distance-d word bigram with distance-d class bigram and find optimal interpolation weights on development data
- Combine using log-linear interpolation the baseline 4-gram, the unigram and the linear interpolated distance-d word and class bigram combination for $d = 3 \ldots 10$ optimising exponents on the development data

The final step above is performed incrementally, starting with the baseline 4-gram model, the unigram and the best distance-3 word and class bigram combination. The optimal weights for the 3 models are found by minimising the perplexity on the development data. Then the best distance-4 word and class bigram combination is added and the optimal weights for all 4 models found using the optimal weights for the first 3 models as the initialisation, and so on. This approach was used to minimise the effect of local minima in the search for an optimal set of weights. The perplexity results on the Penn Treebank evaluation data are given in Table 1 together with the perplexities obtained using two whole-sentence grammar-based models that were linearly interpolated with a word trigram LM in [8, 9].

Model	Perplexity
Chelba & Jelinek	148.9
Charniak	126.1[4]
Baseline 4-gram	126.5
LLI 12-gram	119.9
LLI 12-gram (+WSJ2)	118.2

Table 1. Perplexity on Penn Treebank evaluation data comparing performance of two grammar-based models and a 12-gram log-linear combination of models (LLI).

Model construction for experiments on the Wall Street Journal 5K task were identical except that the baseline LM was a 4-gram word model only and that the number of classes investigated for constructing the distance class models was varied between 200 and 1000 in increments of 200 classes. Perplexity and bigram lattice rescoring results on the Wall Street Journal evaluation data are given in Table 2. This data comprised the male speakers from the Nov'92 and Nov'93 evaluation and development sets (776 sentences, 13113 words).

Model	Perplexity	WER
Baseline 4-gram	50.4	6.21%
LLI 12-gram	45.9	6.10%

Table 2. Perplexity and word error rate from rescoring bigram lattices using a word 4-gram and a 12-gram log-linear combination of models (LLI).

In Figure 1, the ratio of distance bigram perplexities to the unigram perplexity on the development data is given. For the Penn

[2]The authors wish to thank Eugene Charniak for ensuring that the experimental setup was identical to the one he used.

[3]Although this finds the optimal number of classes for use in a stand-alone class model, it is not necessarily the optimal number when combined with other language models.

[4]Perplexity obtained by interpolation at sentence-level rather than at word-level.

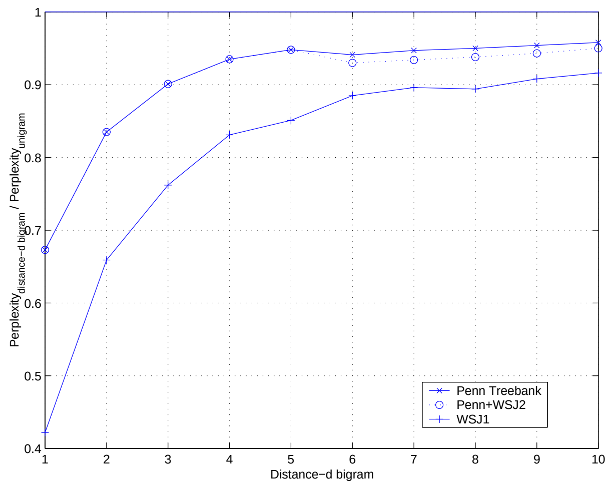

Fig. 1. Variation in perplexity against distance for best combination of word and class distance bigrams computed on Penn Treebank and WSJ1 development data.

Treebank experiments the two sets of results show the ratios both with and without using models built on the additional WSJ2 data.

Using only word-based dependencies the *tree-structure* method was found to be approximately 30 times faster and the *vector of probabilities* method 5 times faster than performing the summation in Equation (6) using un-cached probabilities.

5. DISCUSSION

The first clear observation from the experiments presented above is not a very new one: the choice and construction of the baseline LM can have a large effect on the overall perplexity of the final model and the relative further improvements that are possible. For example, the 4-gram LM that we used had a perplexity that was lower than or close to the perplexities of the whole-sentence grammar-based models that have been reported in the literature.

The second observation is that, given this well-trained baseline LM, subsequent improvements through the inclusion of distance bigram LMs are relatively hard to obtain. Part of the problem was found to be the small amount of training data available for the Penn Treebank experiments. When a much larger amount of training data was used the improvements obtained with the distance bigram information were correspondingly greater. This was observed with the WSJ1 data and also when LMs built on the WSJ2 data were included with the Penn Treebank models.

The third observation is that improvements in the individual component models tend to be carried through when models are combined using log-linear interpolation. This becomes clear when we consider the following perplexity (PP) relationship derived from Equation (2):

$$PP_{LLI} = \frac{1}{PP_Z} PP_{4-gram}^{\lambda_{4g}} \prod_{i=3}^{10} \left\{ \frac{PP_i}{PP_{uni}} \right\}^{\lambda_i} \tag{7}$$

where PP_i is the perplexity of the distance-i bigram. Ignoring the effect of the perplexity of the normalisation term PP_Z, we see that building high quality distance bigram models is very important, even before combination is considered. This is corroborated by the plots in Figure 1. The ratio of distance bigram perplexities to the unigram perplexity gives an indication as to how much more information the distance bigram provides over the unigram model. The lower the ratio the greater the additional information content. For models built exclusively on the Penn Treebank data these ratios were found to be relatively large and significantly higher than those on the WSJ1 data. By including models built on the WSJ2 data these ratios were lowered and greater overall perplexity improvements were obtained.

6. CONCLUSION

In this paper we have described two methods for the efficient computation of the normalisation for log-linear models. This is a particularly important issue for training log-linear models and for using them in decoding or rescoring. Several methods have also been described for constructing high quality distance bigram LMs and the need for large amounts of training data to be available was also demonstrated. It was also shown that perplexity results as good as those using complex whole-sentence grammar-based language models can be obtained using a well-trained 4-gram language model. Further improvements in perplexity of almost 7% were obtained by constructing a log-linear combination of the baseline with a number of linearly interpolated distance word and class bigrams. A similar combination of models on a Wall Street Journal 5K task was shown to give a 9% improvement in perplexity and a 1.8% relative improvement in word error rate over a baseline word 4-gram LM using lattice rescoring.

7. REFERENCES

[1] J. Ponte and W.B. Croft, “A Language Modeling Approach to Information Retrieval,” in *Proceedings of the ACM SIGIR*, 1998, pp. 275–281.

[2] I.G. Varea, F.J. Och, H. Ney, and F. Casacuberta, “Refined Lexicon Models for Statistical Machine Translation using a Maximum Entropy Approach,” in *Proceedings of the 39th Annual Meeting of the ACL*, Toulouse, France, 2001.

[3] R. Rosenfeld, *Adaptive Statistical Language Modelling: A Maximum Entropy Approach*, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, 1994, CMU-CS-94-138.

[4] D. Klakow, “Log-linear Interpolation of Language Models,” in *Proceedings of ICSLP*, Sydney, Australia, 1998.

[5] J. Peters, “Compact Maximum Entropy Language Models,” in *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, Colorado, USA, 1999.

[6] J. Wu and S. Khudanpur, “Efficient Training Methods for Maximum Entropy Language Modelling,” in *Proceedings of ICSLP*, Beijing, China, 2000.

[7] M. Marcus, B. Santorini, and M. Marcinkiewicz, “Building a Large Annotated Corpus of English: the Penn Treebank,” *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1995.

[8] E. Charniak, “Immediate-Head Parsing for Language Models,” in *Proceedings of the 39th Annual Meeting of the ACL*, Toulouse, France, 2001.

[9] C. Chelba and F. Jelinek, “Exploiting Syntactic Structure for Language Modeling,” in *Proceedings of COLING-ACL 98*, New Brunswick, NJ, 1998, pp. 225–231.