SPEED REGULARIZATION AND OPTIMALITY IN WORD CLASSING

Geoffrey Zweig and Konstantin Makarychev

Microsoft Research
{gzweig,komakary}@microsoft.com

ABSTRACT

Word-classing has been used in language modeling for two distinct purposes: to improve the likelihood of the language model, and to improve the runtime speed. In particular, frequency-based heuristics have been proposed to improve the speed of recurrent neural network language models (RNN-LMs). In this paper, we present a dynamic programming algorithm for determining classes in a way that provably minimizes the runtime of the resulting class-based language models. However, we also find that the speed-based methods degrade the perplexity of the language models by 5-10% over traditional likelihood-based classing. We remedy this via the introduction of a speed-based regularization term in the likelihood objective function. This achieves a runtime close to that of the speed based methods without loss in perplexity performance. We demonstrate these improvements with both an RNN-LM and the Model M exponential language model, for three different tasks involving two different languages.

Index Terms— Language Modeling, Word Classes, Recurrent Neural Network, Model M

1. INTRODUCTION

Word classes are used in language modeling for at least two distinct purposes: first, to improve the quality of the resulting language model, and secondly to speed up computationally complex models. In a seminal work, Brown et al. [1] proposed a classing scheme in which each word is assigned to a single class, and the assignment is done in such a way as to maximize data likelihood under a class-based bigram model. This approach was further studied in [2, 3] and developed into a method for optimizing probability under a trigram class model. Later, Goodman [4] observed that class-based models can also be used to speed up otherwise computationally infeasible models, and proposed the use of word classes in maximum entropy language models. The key observation is that a class model works by first computing the probability of the class of a word given some context, and then the probability of the word itself conditioned on the class. The first step requires normalizing over all the classes, and the second step requires normalizing over all the words in a specific class. If we assume that the vocabulary of size V is partitioned into equally likely classes of size $\sqrt{V}$, this reduces the computational complexity from $O(V)$ to $O(\sqrt{V})$. Word classing has since been incorporated in several computationally complex language models: Model M [5, 6] where it achieves both likelihood and speed improvements; and in feedforward and recurrent neural networks [7, 8, 9, 10, 11]. To further improve speed, multi-level classing has been explored in, e.g. [12, 13, 14].

In a novel approach, Mikolov and colleagues [11, 15] abandon the Brown et al. [1] likelihood based classing for a process that groups words of similar frequency together. In a recurrent neural network language model, this classing method has been shown to produce outstanding likelihoods. Following an analysis by Povey, the method was subsequently improved by binning on the basis of the square-root of the frequency rather than the frequency itself [16].

This paper makes three important contributions to the study of word classing. First, we show that the proposed frequency-based classing methods are not in fact speed-optimal, and present a dynamic programming algorithm which is. Secondly, we make a systematic comparison of language models using both frequency and likelihood based classing, and find that likelihood based classing produces models with significantly better perplexity, but also much slower models. Finally, we propose the addition of a *speed-regularization* term to the traditional likelihood objective function, which results in classes that are both computationally efficient and give low perplexities.

The remainder of this paper is organized as follows. In Section 2 we briefly review computation with class based models and identify the computational issues involved. In Section 3, we review previous classing algorithms, and present our new algorithms for speed-optimal and speed-regularized classing. In Section 4, we evaluate the methods with French and English datasets, using both Model M and a recurrent neural network implementation. Section 5 further relates this work to previous efforts, and we conclude in Section 6.

2. COMPUTATION WITH CLASS BASED LANGUAGE MODELS

The basic operation of a class based language model is to break the computation of a word probability into separate class and word based calculations. For computational simplicity, the classing function has traditionally assigned a single class to each word. Denoting a word history as h and the class of a word as $c(w)$, we have

$$P(w|h) = P(c(w)|h)P(w|c(w), h)$$

With this decomposition, separate models may be used to estimate the component probabilities. In maximum entropy and neural network language models the computations involve normalizing over all the labels in the model. In a system that predicts $P(w|h)$ directly, this requires normalizing over all the words in the vocabulary, and is an $O(V)$ operation. In a class model, however, this is broken into two separate normalizations, first over the set of classes $\mathcal{C}$ when computing $P(c(w)|h)$, and secondly over the members of a specific class when computing $P(w|c(w), h)$. Denoting the size of word w's class as $|c(w)|$, the computational complexity of computing $P(w|h)$ is $O(|\mathcal{C}|) + O(|c(w)|)$. Denoting the frequency of word w_i as $f(w_i)$ (computed over a training set with D words), we may now write an expression for the expected computational complexity R of evaluating all the words in the training data using language model that uses a specific classing scheme $\mathcal{C}$.

$$R = D|\mathcal{C}| + D \sum_{c_q \in \mathcal{C}} |c_q|(\sum_{w_j \in c_q} f(w_j))$$

The first term is the contribution of the class part of the model, which must be evaluated for each word. The second term is simply the number of times each class-specific model must be evaluated, multiplied by the number of members of the class. While previously recognized [17], this objective function has not been explicitly optimized in the prior literature, and in Section 3.2, we will present a method for doing so.

3. CLASSING METHODS

3.1. Frequency Bin Classing

In earlier work by Mikolov and colleagues [11, 15], a frequency based classing method was proposed to speed computation with a RNN-LM. In this approach, the words are ordered by frequency. Then, classes are formed from blocks of consecutive words by placing boundaries such that each class accounts for a constant fraction of the total probability mass. This has the property that the lower-numbered classes will have fewer members than higher-numbered classes, because their members are more frequent.

To improve on the runtime performance of frequency based classing, Povey [17] has suggested taking the square-root of the frequencies prior to doing the binning. This has the effect of flattening the distribution, and creating more evenly sized classes.

Data: N: Desired number of classes
V: Number of words in the vocabulary
$cum(p)$: Cumulative frequency of words $w_1 \dots w_p$
$F(k, p)$: Best cost of making k classes ending with word w_p
Initialization: $F(0, 0) = 0$; sort words by frequency
for $k \leftarrow 1$ **to** N **do**
  for $p \leftarrow k$ **to** V **do**
    $F(k, p) = \infty$
    for $s \leftarrow k - 1$ **to** $p - 1$ **do**
      $F(k, p) = \min(F(k, p), F(k - 1, s) + (cum(p) - cum(s)) * (p - s))$
    end
  end
end

Algorithm 1: Recursion to determine optimal classes. Maintenance of standard back-pointers allows recovery of the boundaries.

3.2. Speed Optimal Classing

While frequency based classing provides a large speedup over a purely word-based model, there are no guarantees as to runtime, and it is unclear if the heuristics used are optimal. It is also not obvious whether this is an NP-complete optimization problem, since we have V words which must be shuffled into K classes. In this section, we show that the previous heuristics are not if fact optimal, but that the problem does admit a quadratic time solution, and we present a dynamic programming (DP) algorithm for finding the speed-optimal classing.

The algorithm is based on three observations. First, in the optimal classing, we are free to index the classes in any order; in particular, we may put the class with the fewest members "first"; the next largest class next, and so forth up to the class with the greatest number of members. Second, in the optimal class assignment, it can never be the case that a class with more members contains a word that occurs with greater frequency than any member of a class with fewer members. This is because swapping the two would then improve the objective function. This implies that the words should be sorted by frequency. Finally, the best way of making k classes ending at some position p in the sorted word order must include the best way of making $k - 1$ classes ending at some position less than p. These observations imply that the optimal class assignment can be found by placing the words in order from most to least frequent, and then placing partition boundaries. Thus a DP algorithm which searches over all segmentations of the sorted words will find the optimum. Algorithm 1 makes this explicit with an $O(NV^2)$ procedure. In fact, we can improve the running time to $O(NV \log V)$. Observe that the optimal $s = s(p)$ in Algorithm 1 is a monotone function of p. Thus, once we compute $s(p')$ for some p', we can limit

the search for s for $p < p'$ and $p > p'$ by $s \leq s(p')$ and $s \geq s(p')$ respectively. We replace the "for" loops for p and s by a recursive binary search procedure that considers p's in the order $N/2, N/4, 3N/4$, and so on. Due to space limitations, we omit the details here.

It is worth noting that this algorithm sheds light on another outstanding problem - that of determining how many classes to use. This may be done by computing the optimal way of making up to V classes, and then choosing the number of classes with the global optimum. In fact, the overall objective function is a convex function of the number of classes, so as soon as it begins to increase as the recursion proceeds, the optimum has been found.

3.3. Likelihood Classing

In its original formulation [1, 2], word classing was viewed as a method of improving the perplexity of a language model, and the objective function proposed reflects this. The most commonly used objective function is to maximize the data likelihood under a class based bigram language model:

$$P(w_1 \dots w_n) = \prod_i P(c(w_i)|c(w_{i-1}))P(w_i|c(w_i))$$

Ignoring terms which do not depend on the class assignments, and using c_1c_2 to denote a class bigram occurring in the data, and $N(\cdot)$ the count function, this is equivalent [1, 3] to maximizing

$$\sum_{c_1c_2} N(c_1c_2) \log N(c_1c_2) - 2\sum_c N(c) \log N(c)$$

This objective function can be effectively maximized using a local search procedure and incrementally updating the counts of class bigrams as changes are made. We use a variant of the exchange algorithm of [2] which iteratively sweeps over the vocabulary, and moves each word to a new class so as to greedily improve the objective function.

3.4. Speed Regularized Likelihood Classing

As can be seen, the likelihood objective function is insensitive to the computational complexity induced on a language model by the resulting classes. Furthermore, one might expect that for a large number of words, there are many classes that are about equally good, and an essentially arbitrary decision will be made. Our proposed speed regularization technique makes use of this observation by adding an objective function term which penalizes computational complexity. With speed regularization, combining the objective function of Sections 3.3 and 3.2 and ignoring constants, the objective function becomes:

$$\sum_{c_1c_2} N(c_1c_2) \log N(c_1c_2) - 2\sum_c N(c) \log N(c) \\ -\alpha D \sum_{c_q \in \mathcal{C}} |c_q| (\sum_{w_j \in c_q} f(w_j))$$

Dataset	Train Size	Test Size	Vocab	OOVs
Treebank	0.93M	0.082M	10k	5.8%
Wikipedia	68M	6.1M	126k	2.8%
AFP	47M	1.2M	136k	1.0%

Table 1. Dataset characteristics. Sizes in millions of words.

Here, the speed contribution is weighted by an overall regularization weight α. We have found that a weight of $\alpha = 0.001$ produces good results across a variety of tasks and different numbers of classes.

4. EXPERIMENTAL RESULTS

We tested our classing methods on three datasets. The first is the relatively small Penn Treebank dataset, *Linguistic Data Consortium* catalog number LDC1999T42, normalized as in [10] [1]. The second dataset is a sub-sampling of a Wikipedia dump from January 2011, broken down into sentences, with filtering to remove sentences consisting of URLs and Wiki author comments [18]. The final dataset is a sampling of the French Gigaword corpus, LDC2009T28. We used all the Agency France-Presse data from 2006 and 2007 as training data, with June, 2008 data being used as development data, and August, 2008 as test data. We used the RNN toolkit of Mikolov [16], modified to allow generic classing functions. We restricted the dataset sizes to below 100M words because the RNN training uses non-parallel stochastic gradient descent, and is therefore time-consuming. The data characteristics are summarized in Table 1. In all cases, a separate small validation set was used to control the RNN learning rate; no learning rate or regularization parameters were adjusted for Model M. Four-gram language models trained with the CMU language modeling toolkit achieve perplexities of 162, 190 and 76 respectively.

4.1. RNN Results

Table 2 shows the perplexity of a RNN-LM, using the different classing procedures. The "OS" column presents the results for optimal-speed classing, and "Freq" represents linear frequency binning. The "Povey" column uses the square-root classing scheme. "LL" and "LL+Reg" use an objective function based on the data log-likelihood with and without speed regularization. We see that the perplexities of the speed-based classing methods are similar, and much worse than using likelihood based classing. Most importantly, there is minimal degradation in perplexity introduced via the regularization.

Table 3 shows the run-times of a RNN-LM using the various classing schemes. We see that optimal and Povey classing are essentially the same in speed, and the fastest of the methods. Speed regularization achieves a significant speedup with negligible cost in perplexity. The runtime improvement is similar in both training and test.

[1] We thank T. Mikolov for providing the data.

Dataset	OS	Povey	Freq	LL	LL+Reg
Treebank	132	136	133	**125**	126
Wikipedia	143	143	142	**132**	**132**
AFP	61.4	60.7	60.7	**54.5**	55.5

Table 2. Perplexities of different classing methods with a RNN-LM.

Dataset	OS	Povey	Freq	LL	LL+Reg
Treebank	**0.727**	0.741	1.03	1.04	0.898
Wikipedia	**113**	116	342	382	145
AFP	**68.7**	74.2	259	214	90.0

Table 3. RNN-LM runtime using different classing methods, minutes per sweep through the training data.

In these experiments, we used 100 classes for the Treebank dataset, and 200 classes for the others. We also used trigram max-ent features [15]. The networks all used 50 hidden units.

4.2. Model M Results

To further verify the usefulness of speed regularized classing, we repeated the experiments with an implementation of Model M [5, 6] using 4gm features. These results are interesting from at least two perspectives: first, unlike with the RNN implementation, Model M classing is intended to improve likelihood. Secondly, implementations [19, 20, 21] typically use a set of speedups which depart from the assumptions behind the estimation of computational complexity. Specifically, our implementation operates on N-gram counts (not word-by-word) and the N-grams are sorted in training so the normalizer for each class history is only computed once (not once per word).

The perplexity results of Table 4 show exactly the same trends as the RNN model. Speed-optimal, Povey and Frequency classing all perform about the same, and much worse than likelihood based classing. The use of speed regularization has negligible effect on perplexities.

The runtime results of Table 5 also conform to the RNN pattern. Speed optimal and Povey classing are the fastest, and speed regularization cuts the runtime by 29 and 40% on the larger sets. Table 6 summarizes the results in terms of relative improvements for both Model M and the RNN-LM.

5. RELATION TO PRIOR WORK

Besides the methods mentioned earlier, Chen and Chu [22] have tackled the problem of making better classes for Model M from a likelihood perspective, and there has been recent interest in hierarchical classing methods to improve the runtime of neural network language models [12, 13, 14]. Morin and Bengio [12], propose to represent each word with $O(\log |V|)$ bits, thus implicitly defining a tree with individual words at the leaves. The structure of this tree is inferred from WordNet, and the bits are sequentially predicted on a path from the root to a leaf. Mnih and Hinton [13] use EM to cluster continuous space word representations into a many-level hierarchy, and attempt to maintain approximately balanced structures for computational efficiency. The SOUL architecture [14] also takes a hierarchical prediction approach based on continuous space representations, but in practice adds only one extra layer of prediction beyond the two-level schemes we have focused on. Our work differs from these in adding the exact measure of computational cost to the objective function, and optimizing a combined likelihood and speed objective. Multi-layer hierarchies may benefit from our approach as well.

Dataset	OS	Povey	Freq	LL	LL+Reg
Treebank	154	154	154	**135**	136
Wikipedia	183	183	182	**157**	159
AFP	72.2	72.4	72.3	**62.7**	63.2

Table 4. Perplexities of different classing methods with Model M.

Dataset	OS	Povey	Freq	LL	LL+Reg
Treebank	**1.02**	**1.02**	1.11	1.20	1.14
Wikipedia	142	**129**	209	270	161
AFP	**62.8**	63.0	70.1	106	74.9

Table 5. Model M runtime using different classing Methods, minutes per sweep through the training data.

Dataset	M-PPL	M-RT	RNN-PPL	RNN-RT
Treebank	12%	5.0%	4.5%	14%
Wikipedia	13	40	7.8	62
AFP	12	29	9.6	58

Table 6. Perplexity (PPL) reductions of speed regularization over speed-optimal classing, and runtime (RT) improvement of speed regularization over pure likelihood classing for Model M (M-) and RNN-LM (RNN-) implementations.

6. CONCLUSION

We have seen that likelihood-optimizing classing methods result in classes that induce better language models than speed optimizing methods. To achieve some of the benefits of speed optimizing methods, we propose the addition of a speed regularization term to the traditional likelihood objective function.

In all test conditions, we observe a consistent positive pattern by using speed regularization. The run-times decrease substantially, and the perplexity of likelihood based classing is maintained. Since the use of speed regularization is very simple to implement and imposes essentially no extra overhead in determining the classes, it should certainly be used.

Acknowledgments

We thank Dan Povey and Tomas Mikolov for insightful discussions.

7. REFERENCES

[1] P.F. Brown, V.J. Della Pietra, P.V. deSouza, J. Lai, and R.L. Mercer, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, 1992.

[2] R. Kneser and H. Ney, "Improved clustering techniques for class-based statistical language modeling," in *Proc. Eurospeech*, 1993.

[3] Sven Martin, Jrg Liermann, and Hermann Ney, "Algorithms for bigram and trigram word clustering," *Speech Communication*, vol. 24, no. 1, pp. 19 – 37, 1998.

[4] J. Goodman, "Classes for fast maximum entropy training," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, 2001.

[5] S. Chen, "Performance prediction for exponential language models," in *NAACL-HLT*, 2009.

[6] S. Chen, "Shrinking exponential language models," in *NAACL-HLT*, 2009.

[7] H. Schwenk and J.L. Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. IEEE, 2002, vol. 1, pp. I–765.

[8] Y. Bengio, R. Ducharme, Vincent, P., and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Reseach*, vol. 3, no. 6, 2003.

[9] Holger Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, no. 3, pp. 492 – 518, 2007.

[10] Tomas Mikolov, Martin Karafiat, Jan Cernocky, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Interspeech*, 2010.

[11] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, "Extensions of recurrent neural network based language model," in *ICASSP*, 2011.

[12] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proceedings of the international workshop on artificial intelligence and statistics*, 2005, pp. 246–252.

[13] A. Mnih and G.E. Hinton, "A scalable hierarchical distributed language model," in *Advances in neural information processing systems*, 2009, vol. 21, pp. 1081–1088.

[14] Hai-Son Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Structured output layer neural network language model," in *ICASSP*, 2011.

[15] Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky, "Strategies for Training Large Scale Neural Network Language Models," in *ASRU*, 2011.

[16] Tomas Mikolov, "Rnntoolkit http://www.fit.vutbr.cz/ imikolov/rnnlm/," 2012.

[17] Daniel Povey, ," 2012, personal correspondence.

[18] G. Zweig, J.C. Platt, C. Meek, C.J.C. Burges, A. Yessenalina, and Q. Liu, "Computational approaches to sentence completion," in *Proc. Association of Computational Linguistics*, 2012.

[19] J. Wu and S. Khudanpur, "Efficient training methods for maximum entropy language modeling," in *Interspeech*, 2000.

[20] S. Chen, L. Mangu, B. Ramabhadran, R. Sarikaya, and A. Sethy, "Scaling shrinkage-based language models," in *ASRU*, 2009.

[21] G. Zweig and S. Chang, "Personalizing Model M for Voicesearch," in *ICSLP 2011*, 2011.

[22] S.F. Chen and S.M. Chu, "Enhanced word classing for model m," in *Interspeech*, 2010.