RANDOM CLUSTERINGS FOR LANGUAGE MODELING

Ahmad Emami and Frederick Jelinek

Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218
{*emami, jelinek*}*@jhu.edu*

ABSTRACT

In this paper we present an application of randomization techniques to class-based n-gram language models. The idea is to derive a language model from the combination of a set of random class-based models. Each of the constituent random class-based models is built using a separate clustering obtained via a different run of a randomized clustering algorithm. The random class-based model can compensate for some of the shortcomings of conventional class-based models by combining the different solutions obtained through random clusterings. Experimental results show that the combined random class-based model improves considerably in perplexity (PPL) and word error rate (WER) over both the n-gram and baseline class-based models.

1. INTRODUCTION

The role of a statistical language model is to assign a probability $P(W)$ to any given word string $W = w_1 w_2 \dots w_M$. This is usually done in a left-to-right manner by factoring the probability:

$$P(W)=P(w_1 w_2 \dots w_M)=P(w_1)\prod_{k=2}^{M} P(w_k|W_1^{k-1}), \quad (1)$$

where the sequence of words $w_1 w_2 \dots w_j$ is denoted by W_1^j. Ideally the language model would use the entire history W_1^{k-1} to make its prediction for word w_k. However, data sparseness is a crippling problem with language models and therefore all practical models employ some sort of equivalence classification of histories W_1^{k-1}:

$$P(W)\approx P(w_1)\prod_{k=2}^{M} P(w_k|\Phi(W_1^{k-1})), \quad (2)$$

where $\Phi(W_1^{k-1})$ denotes the class of the word string $(w_1 \dots w_{k-1})$. However in most cases, even after using an equivalence classification $\Phi(\cdot)$, there is still not enough data to reliably estimate the conditional probabilities $P(w_k|\Phi(W_1^{k-1}))$. Therefore, proper smoothing techniques need to be used to make sure that the events not observed or rarely seen in the training data will be assigned appropriate non-zero probabilities.

Research in statistical language modeling is in general concerned with two fundamental problems: finding compact and yet specific enough classification schemes, and developing powerful smoothing (generalization) techniques. For a good review of different language modeling approaches see [1].

The most widely used language models are the so called n-gram models where a word string W_1^{k-1} is classified into word substring W_{k-n+1}^{k-1}. The n-grams models perform surprisingly well given their simple structure, but lack the ability to use longer histories for word prediction, and they still suffer from severe data sparseness problems.

Class-based n-gram models have been shown to be helpful in fighting the data sparsity problem. Their ease of use in implementation and deployment (e.g. in a speech recognizer's decoder) makes them one of the most commonly used models employed to improve upon standard n-gram models. However, due to the shortcomings of the existing clustering algorithms as well as the non-optimal assumptions made in the modeling, any class-based language model can be thought of as a sub-optimal or 'weak' model.

In this paper we propose a random class-based language model and investigate the use of randomization for class-based n-gram models. The approach here is similar to that in Random Forests [2, 3], although that work is in the context of decision trees. Our proposed model is built as a combination of multiple randomized class-based models where each of the individual random models is constructed using a different random clustering. To obtain the random clusterings we propose a few randomization techniques to apply to a word clustering algorithm. By measuring their distance from each other, the resulting random clusterings are shown to be indeed different. Experimental results show that the combined random class-based LM improve considerably in perplexity (PPL) and word error rate (WER) over both the baseline word and class-based n-grams models.

This paper is organized as follows: In section 2 we describe the class-based language models, briefly discussing some clustering algorithms and a few modeling options, emphasizing the exchange algorithm which is our clustering algorithm of choice throughout the paper. In section 3, randomization techniques for the clustering algorithm are proposed and described. Experimental results are given in section 4 followed by discussion and a proposal of future work.

2. CLASS-BASED N-GRAM MODELS

As mentioned in the previous section, one way to fight the data sparsity of n-gram models is to use word equivalence classes. In this framework, the history classification adopts the same Markovian assumption as the n-gram models; that is the probability of the current word w_k depends on only the last $n-1$ previous words. The generalization (smoothing) is achieved through the use of classes. The intuition is that even if a certain word m-gram does not appear in the training data, it is still quite likely that the class m-gram is observed in the same data.

The word classification can be many-to-one or many-to-many, alternatively referred to as hard and soft clustering respectively. This paper deals only with the hard clustering case, where each

This work was supported by the National Science Foundation under grant No. IIS-0085940.

word belongs to one and only one class. Aside from the hard or soft clustering choice, there are two major issues in building class-based LMs. The first issue is how the word classes are derived, and the second one concerns the choice of the model structure given the derived word classes.

There are a variety of methods for obtaining word classes. An obvious choice is to use the syntactic category, or part of speech (POS) tag of words to group them into classes. Alternatively, one can obtain the word classes in a data-driven way using a statistical criterion, which in most cases (but not all) is the likelihood of the training data. Statistically derived classes have been shown to result in better class-based language models than the POS based clusterings. There are quite a few different statistical algorithms for building the word classes. In [4], an agglomerative algorithm is used where class pairs are merged one at a time, with the starting point being each word defining its own class. There are many other statistical methods to derive word classes, discussion of all of which is not in the scope of this paper.

In [5, 6], a non-hierarchical method, called the *exchange algorithm*, is used to find the word classes maximizing the training data likelihood. In our preliminary experiments we found the exchange algorithm to perform no worse than the bottom-up maximum likelihood based method described in [4]. In fact, in addition to being computationally more expensive, the bottom-up method still needs the exchange algorithm to be performed on its final classes (nodes) in order to achieve its best performance.

In the exchange algorithm it is assumed that the number of classes are known and so there is a fixed set C of classes forming a partitioning of the vocabulary V. The goal is to find a many-to-one mapping $\mathcal{M} : V \rightarrow C$ that maximizes some given criteria, which in our case is the likelihood of a given *clustering data.* Note that the clustering data does not have to be the same as the training data later used to estimate the class-based models. The exchange algorithm reaches a local optimum by looping through each word w in the vocabulary V, moving it tentatively to all the classes in C, and assigning it to that class that results in the highest log-likelihood of the clustering data. The procedure is then repeated until convergence is reached.

We should note here that most class-based language models in the literature share the same mapping $\mathcal{M}$ for all the positions in the n-gram. In the few cases that mappings are different (e.g. [1]), they are only distinguished by predicting and predicted positions, and furthermore the position dependent mappings are found separately and independently of each other. It is also the case that most clustering techniques use only bigram statistics to obtain the classes ([6] is one exception). In our work we use the very general case of trigram clustering where trigram statistics are used to find three separate clusterings simultaneously. Algorithm 1 shows the generalized exchange algorithm where three separate mappings $\mathcal{M}^0, \mathcal{M}^1$, and $\mathcal{M}^2$ are used for the predicted position and the two predicting positions respectively. Unlike previous efforts in position dependent clustering, the algorithm derives the three mappings simultaneously; at any point in the algorithm, the update of a given mapping (e.g. $\mathcal{M}^1$) is dependent on the current state of the two other mappings ($\mathcal{M}^0, \mathcal{M}^2$). The log-likelihood of the clustering data, maximized in step 7 of the algorithm, is given by:

$$\sum_{k=1}^{M} log\left(P(C_k^0|C_{k-2}^2, C_{k-1}^1)P(w_k|C_k^0)\right), \quad (3)$$

where $C_j^i = \mathcal{M}^i(w_j)$. To the best of our knowledge this is the first time trigram statistics are used to find position dependent clusterings. Note that one advantage of this generalized clustering is that it can be used for smoothing any conditional probability model

Algorithm 1 Exchange Algorithm

1: Decide on a fixed number of classes for each position
2: Start with initial mappings $\mathcal{M}^i : V^i \rightarrow C^i,\ i = 0, 1, 2$
3: **for each** iteration **do**
4: **for each** word position $i = 0, 1, 2$ **do**
5: **for each** word $w \in V^i$ **do**
6: move w tentatively to all classes in C^i
7: assign w to class c maximizing likelihood

where in general the vocabularies for different positions are not the same.

Once the classes are obtained, they can be used in a variety of methods to build a class-based language model. One model structure commonly used is of the form $P(w_k|w_{k-2}, w_{k-1}) \approx P(C_k^0|C_{k-2}^2, C_{k-1}^1)P(w_k|C_k^0)$ (see [4, 6]), which is in fact the model structure assumed in computing the log-likelihood in Equation 3. In [1], different class-based language models are examined and it is shown that the model above does not produce the best perplexity. Our preliminary experiments confirmed the same and therefore we opt to use the following model throughout this paper:

$$P(w_k|w_{k-2},w_{k-1}) \approx P(C_k^0|w_{k-2},C_{k-2}^2,w_{k-1},C_{k-1}^1) \times P(w_k|w_{k-2},C_{k-2}^2,w_{k-1},C_{k-1}^1,C_k^0). \quad (4)$$

3. RANDOM CLUSTERINGS

As with any other word clustering algorithm, the exchange algorithm is greedy in general and finds a clustering that is only locally optimal. Also, the assumptions made in the modeling structure are usually non-optimal and hence lead to a degradation of the model. One such assumption for example is that each word belongs to only one class which contradicts the fact that a word can have different syntactic or semantic roles depending on the context. Given these shortcomings, we can think of any class-based language model of general from in Section 2 as an sub-optimal or *weak* model. In the spirit of work in [2, 3], it should then be possible to combine many random weak clusterings to obtain a more powerful model.

In this paper we obtain random class-based models by randomizing the clustering algorithm while keeping the modeling structure (Equation 4) fixed. We propose the following methods for randomizing the exchange algorithm:

- *random clustering data* In this randomization technique, a random subset of the original clustering data is used for each random clustering set. In our experiments we use a uniform distribution to randomly sample (with replacement) trigram tokens from the original data until the sampled data is of the same size as the original data. Note that unlike the other randomization methods in this paper, random data selection is not specific to the exchange algorithm and can be applied to any other clustering or modeling technique.
- *random vocabulary limitation* In this method, the participating words in the exchange algorithm are limited to a random subset of the vocabulary at the start of each iteration and for the duration of that iteration, only the words belonging to the random subset are allowed to change class memberships. This can be thought of as replacing V^i at step 5 of Algorithm 1 with a randomly chosen subset of a given size. In our experiments we use the uniform distribution to sample random subsets half the size of the original

vocabulary. It is easy to see that limiting the vocabulary leads to a linear decrease in the complexity of each iteration and since experiments show that the total number of iterations does not increase, this method can also be used for speeding up the clustering algorithm.

- *random initialization* In this method, the initial mappings $\mathcal{M}^i$ in step 2 of the algorithm are randomly chosen. The random initialization works by assigning each word in the vocabulary to a randomly chosen class through a uniform distribution.
- *combination* All three randomization techniques above are used in conjunction with each other.
- *random number of classes* Instead of deciding on a fixed number in step 1 of the algorithm, the number of classes can be chosen randomly so that each clustering will correspond to a partitioning of the vocabulary with different number of classes.

The final random class-based model is formed as a linear combination of the individual class-based models built using each of the random clusterings:

$$P(w_k|w_{k-2}, w_{k-1}) = \sum_{r=1}^{R} \lambda_r P_r(w_k|w_{k-2}, w_{k-1}) \qquad (5)$$

where $P_r(w_k|w_{k-2}, w_{k-1})$ is an individual class-based model in the form of Equation 4, λ_r is the correspoding interpolation weight, and R is the number of random clusterings.

4. EXPERIMENTS AND RESULTS

Our experimental setup is as follows: for perplexity results we used the UPenn Treebank portion of the WSJ corpus. The corpus is divided into training, held-out, and test sets containing 930k, 74k, and 82k words respectively. We used an open vocabulary consisting of 10k words. The clustering data consists of the training data plus the held-out data.

The WER experiments consisted of the re-scoring of the N-best lists output by a speech recognizer. We evaluated our models in the WSJ DARPA'93 HUB1 test setup with a 20k open vocabulary. The test set is a collection of 213 utterances for a total of 3446 words. The trigram language model used by the speech recognizer to obtain the N-best lists was trained on 40M words. However, all our models are trained on a 19M subset of this data. Interpolated Kneser-Ney smoothing was used for all our models in this work.

In order to show that randomization can improve over even the best individual class-based model, we proceed by finding the best number of classes for each position. For this purpose we took the exhaustive approach of trying many different partition sizes. In the UPenn perplexity setup, we considered 64, 128, 256, 512, 1024, 2048, 4096, and 10000 classes for the predicting positions _1 and _2, and 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 10000 classes for the predicted position _0. Of all the $8 \times 8 \times 10 = 640$ configurations, the $|C^2| = |C^1| = 10000, |C^0| = 64$ setup gave the best test set perplexity. The same configuration is also used for the WER experiments. Note that in this setup there are no clusterings in positions _2 and _1 and the class-based model in Equation 4 in reduced to $P(C_k^0|w_{k-2}, w_{k-1}) \times P(w_k|w_{k-2}, w_{k-1}, C_k^0)$.

Table 1 shows the perplexity results for the random class-based model. Both training and held-out data were used for estimating the class-based models for getting the test set results, while only the training data was used for the held-out perplexities. Each randomization technique was used 100 times to give a set of

random tech.	held-out	test	held-out	test
3-gram	-	-	162	146
baseline	-	-	146.1	137.7
data	149.5±0.14	139.7±0.12	138.4	130.4
vocab limit	146.4±0.19	137.6±0.09	137.8	129.7
initialization	146.4±0.11	137.5±0.12	**136.9**	**128.8**
comb. (all above)	149.3±0.15	139.4±0.12	137.3	129.2
num classes	146.6±0.12	137.8±0.16	137.8	129.8

Table 1. UPenn Perplexity

model	*intp. weight*				
	1.0	0.8	0.6	0.4	0.2
3-gram	13.9	13.9	13.8	13.8	13.4
class	13.3	13.3	13.1	13.4	13.3
random class	13.0	12.9	12.9	12.8	13.2

Table 2. WSJ Word error rate

100 clusterings. Each of the random clusters was then used independently to build a class-based model according to the equation given above. The first two columns show the individual random clustering perplexities for the different randomization techniques. For example, 141.4 ± 0.13 denotes that the 100 random clusters formed by random initialization have in average a perplexity of 141.4 with a sample variance of 0.13. The rows 'data', 'vocab limit', and 'initialization' refer to the random data, random vocab limitation, and random initialization methods respectively. The row 'comb.' denotes the case where all the aforementioned methods are used concurrently and the row 'num classes' shows the results for the random number of classes randomization technique where the number of classes were drawn from a normal distribution with mean 64 and standard deviation 16. To combine the 100 individual class-based models, they were linearly interpolated with equal weights ($\lambda_r = 0.01$). The results are shown in the last two columns corresponding to held-out and test sets respectively. Finding the interpolation weights on the held-out set did not result in an improvement in test set perplexity. Furthermore, interpolating the final combined model with the word trigram did not result in any perplexity improvement over the combined model, showing that unlike conventional class models, the random class-based LM does not benefit from the word n-gram model.

The N-best re-scoring WER results are given in Table 2. The three rows refer to word trigram, baseline class-based, and random class-based models respectively. A set of 100 random clusterings were obtained using the random initialization technique and the final model was again in the form of Equation 5 (with $\lambda_r = 0.01$). Each of the models is interpolated with the original N-best LM scores at different values of interpolation weight. The original N-best list by itself (intp. weight=0.0) will give a WER of 13.7%. It can be observed that the random class-based models consistently outperforms both the trigram and baseline class-based models.

4.1. Clustering Distances

In order to show that the clusterings obtained through the separate runs of the randomized exchange algorithm are indeed different, we use two clustering comparison methods to measure the distance between the random clusterings.

The first of these measures, called *Classification Error* (CE), is defined as the minimum number of words that have to change membership in one clustering to make it equal to the other one [7]. Two clusterings are defined to be equal if they form the same partitioning over the vocabulary [7].

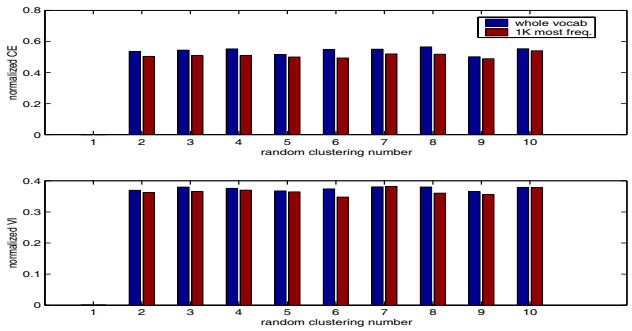

Fig. 1. Class distance across random clusterings

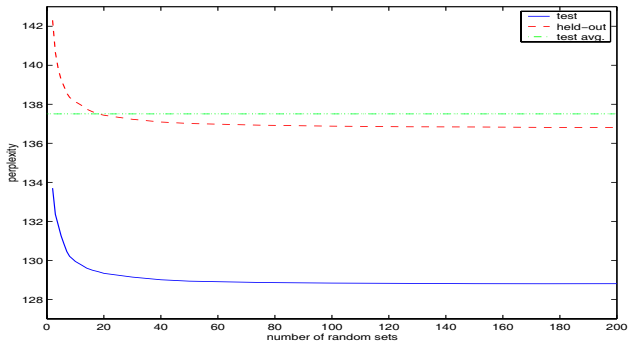

Fig. 2. PPL evolution with increasing num of random sets

The second measure uses an information theoretic criterion called *Variation of Information* (VI), to define the distance between a pair of clusterings as the amount of information lost or gained when changing from one clustering to the other [8]. The VI distance is bounded from above by $log(|V|)$. Both the CE and VI distances are symmetric and transitive, and are true metrics on the space of clusterings.

Figure 1 shows the normalized CE and VI distances between the first random clustering set with the the first 10 sets. The clusters are from the UPenn perplexity setup and were created using the random initialization method. Since the exchange of rare words among classes doesn't change the likelihood significantly, it is quite possible that the rare words were assigned *arbitrarily* (without any statistical significance) to different classes in different random clusterings. To make sure that the random clustering are in fact different, the class distances were computed for only the 1000 most frequent words as well. The CE and VI distance measures are normalized by dividing them by vocab size $|V|$ and VI upper bound $log(|V|)$ respectively, ensuring that the distances for different (whole and 1K most frequent) vocabs are comparable. The average normalized CE and VI distance between the possible $\binom{100}{2}$ pairs of the 100 random clusterings were found to be 0.53 and 0.37 respectively. As can be seen, in either whole vocabulary or just the most 1000 frequent words cases, the randomization results in clusterings that are quite different from each other.

5. DISCUSSION AND FUTURE WORK

We have developed a randomized class-based n-gram model which improves considerably in perplexity and WER over both the baseline word n-gram and class-based models. It has been shown that randomization results in clusterings that are quite different from each other. Combining the resulting class-based models resulted in considerable improvements in the performance, showing that the different clusterings represent uncorrelated interpretations of the data. We tend to think of each of the individual random clusterings as a different local optimum of the clustering criterion. As a result, the randomization compensates for the greedy clustering algorithm and the rigid class-based model structure by allowing multiple solutions to the clustering problem. For example the problem of having a given word assigned to only one class is partly overcome, since the word can take different class membership in different random clusterings.

Figure 2 shows how the UPenn test perplexity changes as we use more random class-based models. The clusterings were obtained using the random initialization method. It can be seen that perplexity tapers off very fast and that it is possible to get very good performance with just 20 random sets.

It is interesting to note that all randomization techniques achieve more or less the same result. This further supports the argument above that randomization works by providing different local optimum solutions and that it doesn't matter how these different points are reached.

An interesting observation is that, for a given randomization method, all the individual random models have almost the same perplexity, with a very small sample variance across them. A simple explanation is that all the local optimum points lie at the same level in the log-likelihood surface. However, we believe that this observation requires further investigation.

While the simple interpolation method we used in combining the different random clusterings resulted in significant improvements, we believe better results can be achieved by using better combination methods, either at word probability model (e.g. log-linear models), or by combining all the different clusterings to build a single, though more complex, class-based language model.

6. REFERENCES

[1] Joshua Goodman, "A bit of progress in language modeling," Tech. Rep. MSR-TR-2001-72, Machine Learning and Applied Statistics Group, Microsoft Research, Redmond, WA, 2001.

[2] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, , no. 9, pp. 1545–1588, 1997.

[3] Leo Breiman, "Random forests," Tech. Rep., Statistics Department, University of California, Berkeley, Berkeley, CA, 2001.

[4] P. F. Brown, V. J. Della Pietra, P. V. de Souza, J. C. Lai, and R. L. Mercer, "Class based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

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

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

[7] A. Emami, "A class distance algorithm," January 2003, CLSP Research Note.

[8] Marina Meilă, "Comparing clusterings by the variation of information," in *COLT*, 2003.