BAYESIAN CLASS-BASED LANGUAGE MODELS

Yi Su

Nuance Communications, Inc.
1500 rue University, suite 935, Montréal, Québec H3A 3S7, Canada
Yi.Su@nuance.com

ABSTRACT

By capturing the intuition of "similar words appear in similar context", the Class-based Language Model (CLM) has found success from research projects to business products. However, most CLMs make a simplifying assumption that one word belongs to one class, which models poorly the fact that many words have multiple senses thus should belong to multiple classes.

We propose a Bayesian formulation of the CLM, where a many-to-many mapping between words and classes, i.e., soft-clustering, are naturally supported. A simple collapsed Gibbs sampler is provided to carry out the inference. Not only did we achieve a 22% relative reduction in perplexity on a Wall Street Journal corpus, but also reduced the word error rate of a state-of-the-art conversational telephony speech recognizer by 6% relative.

Index Terms— class-based language model, Bayesian statistics, Gibbs sampling, hierarchical Pitman-Yor process

1. INTRODUCTION

The term *"class-based language model"* refers to not one, but a family of language models which make use of word classes to improve their performance. Since [1], a lot of research has been done to build more sophisticated model and/or find better word classes.

Most CLMs maintain a *hard-clustering* assumption that one word belongs to one class. While it might simplify the problem and reduce the computational cost, it certainly fails to account for the fact that one word can have multiple senses. *Soft-clustering*, where one word belongs to many classes with probability, for bi-gram language model was explored in [2] using Expectation-Maximization.

Bayesian methods have been applied to language modeling as early as in [3] but did not gain popularity until a Bayesian language model based on Pitman-Yor process with state-of-the-art performance was introduced in [4].

The closest previous work to ours is a bi-gram version of Latent Dirichlet Allocation (LDA) [5], which assumed that the corpus was organized in documents. While "document" is a natural concept for topic modeling or information retrieval, we argue that it is neither natural, nor always available, in language modeling.

The rest of the paper is organized as follows: Section 2 recaps the classical hard-clustering CLM. In Section 3, we provide the formal definition of, and an inference algorithm for, a soft-clustering CLM. In Section 4 we present a fully Bayesian CLM based on hierarchical Pitman-Yor process. Experimental setups and results are presented in Section 5. We conclude in Section 6.

Many thanks to Max Bisani, Ernie Pusateri, Wilson Tam and Paul Vozila for stimulating discussions.

2. CLASS-BASED LANGUAGE MODELS

2.1. Model definition

In this paper, the **Class-based Language Model** is defined as

$$P(w \mid h) = P(c(w) \mid h) P(w \mid c(w), h), \tag{1}$$

where $c(w)$ is the unique class to which the word w belongs and h is the conditioning history. We call $P(c(w) \mid h)$ the **class model** and $P(w \mid c(w), h)$, the **word model**.

2.2. Finding classes

Two popular algorithms for finding the word classes are agglomerative clustering [1] and exchange-based clustering [6]. The two methods have their own pros and cons but a carefully implemented exchange-based clustering usually runs much faster than its agglomerative counterpart does.

2.3. Random clustering

Inspired by the random forest language model [7], [8] successfully applied the idea of randomization to the CLM. By averaging a number of CLMs with word classes derived from randomly initialized exchange-based clustering, their model becomes

$$P(w \mid h) = \frac{1}{K} \sum_{k=1}^{K} P_k(c_k(w) \mid h) P_k(w \mid c_k(w), h), \tag{2}$$

where K is the number of CLMs and $c_k(w)$ is the unique class to which the word w belongs in the k-th CLM.

3. SOFT-CLUSTERING CLMS

3.1. Model definition

The **Soft-clustering Class-based Language Model** (SCLM) is defined as

$$P(w \mid h) = \sum_{c \in \mathcal{C}} P(c \mid h) P(w \mid c, h), \tag{3}$$

where $\mathcal{C}$ is the set of all word classes.

3.2. Random sampling

Given a class assignment of all words in the training set, we can estimate the class model $P(c \mid h)$ and the word model $P(w \mid c, h)$ as a regular n- and $(n+1)$-gram LM with Kneser-Ney smoothing and compute the probability given that particular class assignment as

$$P_A(w \mid h) = \sum_{c \in \mathcal{C}} P_A(c \mid h) P_A(w \mid c, h), \tag{4}$$

where A denotes a class assignment for all words in the training set. However, the number of possible class assignments is $|\mathcal{C}|^N$, where N is the number of words in the training set and $|\mathcal{C}|$ is the number of classes. We cannot enumerate this huge space so we take samples from it. Therefore our model becomes

$$\begin{aligned} P(w \mid h) &= \sum_{A \in \mathcal{A}} P(A) \sum_{c \in \mathcal{C}} P_A(c \mid h) P_A(w \mid c, h) \\ &\approx \frac{1}{K} \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} P_{A_k}(c \mid h) P_{A_k}(w \mid c, h), \end{aligned} \tag{5}$$

where K is the number of samples and $\mathcal{A}$ is the set of all possible class assignments.

3.3. Inference algorithm

The key to the SCLM is to draw samples of class assignments. We use a collapsed Gibbs sampler [9], which is an instance of Markov Chain Monte Carlo. Let c_i be the class assignment of the i-th word in the training corpus, w_i.

$$\begin{aligned} P(c_i = j \mid \mathbf{c}_{\neg i}, \mathbf{w}) &\propto P(c_i = j \mid \mathbf{c}_{\neg i}, \mathbf{w}_{\neg i}) \\ &\cdot P(w_i \mid c_i = j, \mathbf{c}_{\neg i}, \mathbf{w}_{\neg i}), \end{aligned} \tag{6}$$

where $\mathbf{c}_{\neg i}$ is the set of class assignments except c_i and $\mathbf{w}_{\neg i}$ is the set of training word tokens except w_i. The first term of the right-hand side of Equation 6 can be computed with a "leave-one-out" version of our class model $P_{\neg i}(c \mid h)$ and the second term can be computed with a "leave-one-out" version of our word model $P_{\neg i}(w \mid c, h)$. Our Gibbs sampling procedure is outlined in Algorithms 1 and 2.

```
Input: Training ngrams $W = \{(h, w)\}$, counts $t(h, w)$
Output: Class assignment $A$, discounts $D_c$ and $D_w$
1  begin
2      $A \leftarrow$ sampleClusters($W$);
3      Update counts for class model with $A$;
4      $D_c \leftarrow$ estimateDiscounts(class model);
5      Update counts for word model with $A$;
6      $D_w \leftarrow$ estimateDiscounts(word model);
7      Return $A$, $D_c$ and $D_w$;
8  end
```

Algorithm 1: sampleGibbsOnce(W)

```
Input: Training ngrams $W = \{(h, w)\}$, counts $t(h, w)$
       and current corpus class assignment $A_c$
Output: Class assignment $A$
1  begin
2      $A \leftarrow \phi$;
3      foreach $h \in \{h' : \exists w, (h', w) \in W\}$ do
4          foreach $w \in \{w' : (h, w') \in W\}$ do
5              foreach $j \in \mathcal{C}$ do
6                  $\theta_j \leftarrow P(c(w) = j \mid A_c, W)$;
                   // by Equation 6
7              end
8              for $i \leftarrow 1$ to $t(h, w)$ do
9                  $c(w_i) \sim$ Multinomial($\{\theta_j\}$);
10                 $A \leftarrow A \cup \{c(w_i)\}$;
11             end
12         end
13     end
14     Return class assignment $A$;
15 end
```

Algorithm 2: sampleClusters(W)

The procedure "estimateDiscounts()" is a widely used method of estimating Kneser-Ney smoothing discounts based on "counts of counts" [10]:

$$d = \frac{n_1}{n_1 + 2n_2}, \tag{7}$$

where n_1 and n_2 are the numbers of n-grams appear in the training set exactly once and twice, respectively.

4. FULLY BAYESIAN FORMULATION

4.1. Hierarchical Pitman-Yor language model

As a prerequisite, we recapitulate the Hierarchical Pitman-Yor Language Model (HPYLM) [4], a Bayesian language model assuming the following generative process:

1. Choose parameters e_j and μ_j for $j \in \{0, 1, \cdots, n-1\}$:

$$\begin{aligned} e_j &\sim \text{Beta}(1,1) \\ \mu_j &\sim \text{Gamma}(1,1) \end{aligned} \tag{8}$$

2. For every word w_i and its history $h_i = w_{i-1} \cdots w_{i-n+1}$:

$$\begin{aligned} H_\phi(w) &\sim \text{PY}(e_0, \mu_0, H_0(w)) \\ H_{w_{i-1}}(w) &\sim \text{PY}(e_1, \mu_1, H_\phi(w)) \\ H_{w_{i-1}w_{i-2}}(w) &\sim \text{PY}(e_2, \mu_2, H_{w_{i-1}}(w)) \\ &\cdots \\ w_i \mid h_i &\sim \text{Mult}(H_{h_i}(w)). \end{aligned} \tag{9}$$

$H_0(w)$ is a uniform distribution over the vocabulary. PY$(\cdot)$ stands for a Pitman-Yor process; Mult$(\cdot)$ stands for a multinomial distribution. [4] convincingly showed that an n-gram LM with interpolated Kneser-Ney smoothing could be regarded as an approximation to the HPYLM, both in theory and in practice.

4.2. Class-based hierarchical Pitman-Yor LM

Having established the HPYLM as the Bayesian counterpart of Kneser-Ney smoothing, we can have a Bayesian version of the CLM by estimating the class and word models with HPYLMs, instead of Kneser-Ney smoothing. We call this model the **Class-based Hierarchical Pitman-Yor Language Model** (CHPYLM).

4.3. Soft-clustering class-based HPY LM

The **Soft-clustering Class-based Hierarchical Pitman-Yor Language Model** (SCHPYLM) assumes the following generative process:

1. Choose parameters d_j and θ_j for $j \in \{0, 1, \cdots, n-1\}$:

$$\begin{aligned} d_j &\sim \text{Beta}(1,1) \\ \theta_j &\sim \text{Gamma}(1,1) \end{aligned} \tag{10}$$

2. Choose parameters e_j and μ_j for $j \in \{0, 1, \cdots, n\}$:

$$\begin{aligned} e_j &\sim \text{Beta}(1,1) \\ \mu_j &\sim \text{Gamma}(1,1) \end{aligned} \tag{11}$$

3. For every word w_i and its history $h_i = w_{i-1} \cdots w_{i-n+1}$:

   (a) Generate c_i:

$$\begin{aligned} G_\phi(c) &\sim \text{PY}(d_0, \theta_0, G_0(c)) \\ G_{w_{i-1}}(c) &\sim \text{PY}(d_1, \theta_1, G_\phi(c)) \\ G_{w_{i-1}w_{i-2}}(c) &\sim \text{PY}(d_2, \theta_2, G_{w_{i-1}}(c)) \\ &\cdots \\ c_i \mid h_i &\sim \text{Mult}(G_{h_i}(c)), \end{aligned} \tag{12}$$

   (b) Generate w_i:

$$\begin{aligned} H_\phi(w) &\sim \text{PY}(e_0, \mu_0, H_0(w)) \\ H_{c_i}(w) &\sim \text{PY}(e_1, \mu_1, H_\phi(w)) \\ H_{c_i w_{i-1}}(w) &\sim \text{PY}(e_2, \mu_2, H_{c_i}(w)) \\ H_{c_i w_{i-1} w_{i-2}}(w) &\sim \text{PY}(e_3, \mu_3, H_{c_i w_{i-1}}(w)) \\ &\cdots \\ w_i \mid c_i, h_i &\sim \text{Mult}(H_{c_i h_i}(w)). \end{aligned} \tag{13}$$

$G_0(c)$ is a uniform distribution over the set of classes; $H_0(w)$, over the set of words, i.e., the vocabulary.

The collapsed Gibbs sampler for the SCLM can be easily adapted to the SCHPYLM by replacing the routine to estimate Kneser-Ney discounts for a routine to sample the parameters of the HPYLM, such as the Gibbs sampler described in [11].

5. EXPERIMENTS

5.1. Perplexity results

We used Wall Street Journal portion of the Penn Treebank and preprocessed the text by lowercasing words, removing punctuations and replacing numbers with the "N" symbol. Sections 00-22 (1,003,324 words) and sections 23-24 (82,430 words) were used as training and testing sets, respectively. The vocabulary size was 10k, including a special token for unknown words.

The orders of our models were all 3. After building a baseline 3-gram LM with Kneser-Ney smoothing, we built CLMs and SCLMs with the numbers of classes being exponents of 2. Then we built Bayesian counterparts of those models by replacing Kneser-Ney n-gram LMs with HPYLMs, as described in Section 4. For each experiment, 100 samples and 800 iterations of burn-in are used. We plotted the results in Figure 1, where the x-axis of the plot is in log-scale, and summarized them in Table 1.

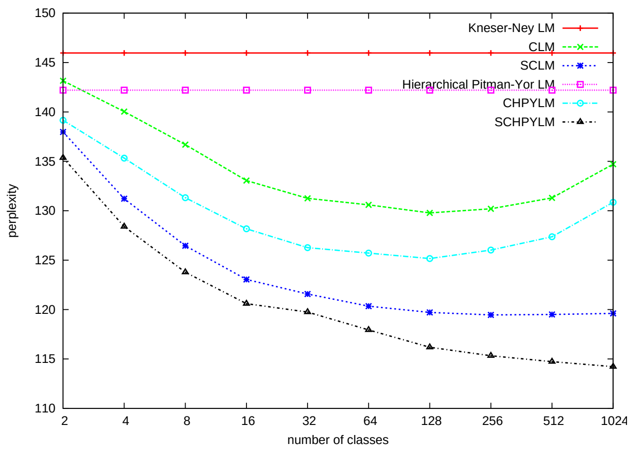

Fig. 1. Perplexity as function of the number of classes

Model	Best Config.	Perplexity
Kneser-Ney	–	146.0
CLM	128 classes	129.8
SCLM	256 classes	119.5
Hier. Pitman-Yor	–	142.2
CHPYLM	128 classes	125.2
SCHPYLM	1024 classes	**114.2**

Table 1. Perplexity result highlights

Model	w/o Interp.	w/ Interp.
Kneser-Ney	14.4	13.5
CLM	14.2	13.4
SCLM	13.7	13.3
Hier. Pitman-Yor	14.1	13.4
CHPYLM	13.7	13.2
SCHPYLM	**13.5**	**13.1**

Table 2. Lattice-rescoring WERs on IBM RT-04 CTS

5.2. Word error rates

For speech recognition results, we used a lattice-rescoring setup with the IBM 2004 Rich Transcription (RT-04) Conversational Telephony Speech (CTS) system [12]. We trained all models with 4-grams, 16 classes, 50 samples and 100 iterations of burn-in on 20M words of Fisher data. The test set was the 2004 development data (DEV04) with 38K words. The vocabulary size was 30K.

We rescored the lattices generated from the first pass decoding of the IBM RT-04 CTS system using models built from Fisher data alone, as well as interpolating them with a big LM built from many other corpora, including a 525M words of "Fisher-like" web data collected by the University of Washington.

As shown in Table 2, SCHPYLM was able to reduce the WERs by 0.9% absolute without interpolation and 0.4% absolute with interpolation. Both reductions were statistically significant with $p < 0.001$.

6. CONCLUSIONS

We proposed the Soft-clustering Class-based Hierarchical Pitman-Yor Language Model (SCHPYLM), a fully Bayesian class-based language model, and provided a collapsed Gibbs sampler to carry out the inference. The perplexity results on the Wall Street Journal part of Penn Treebank were among the best ones previously reported. Lattice rescoring on the IBM RT-04 CTS system with a SCHPYLM resulted in a statistically significant improvement.

7. REFERENCES

[1] P. F. Brown, V. J. D. Pietra, P. V. deSouza, 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.

[2] L. Saul and F. Pereira, "Aggregate and mixed-order Markov models for statistical language processing," in *Proceedings of the Conference on Empirical methods in natural language processing*, 1997.

[3] D. J. C. MacKay and L. C. B. Peto, "A hierarchical Dirichlet language model," *Natural Language Engineering*, vol. 1, no. 3, pp. 1–19, 1994.

[4] Y. W. Teh, "A hierarchical Bayesian language model based on Pitman-Yor processes," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 2006, pp. 985–992.

[5] H. M. Wallach, "Topic modeling: beyond bag-of-words," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 977–984.

[6] S. Martin, J. Liermann, and H. Ney, "Algorithms for bigram and trigram word clustering," *Speech Communication*, vol. 24, no. 1, pp. 19–37, 1998.

[7] P. Xu and F. Jelinek, "Random forests in language modeling," in *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, D. Lin and D. Wu, Eds. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 325–332.

[8] A. Emami and F. Jelinek, "Random clusterings for language modeling," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, March 2005, pp. 581–584.

[9] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, pp. 5228–5235, April 2004.

[10] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependencies in stochastic language modelling," *Computer Speech and Language*, vol. 8, pp. 1–38, 1994.

[11] Y. W. Teh, "A Bayesian interpretation of interpolated Kneser-Ney," School of Computing, National University of Singapore, Tech. Rep. TRA2/06, 2006.

[12] H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig, "The IBM 2004 conversational telephony system for rich transcription," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005, pp. 205–208.