Hierarchical Bayesian Language Models for Conversational Speech Recognition

Songfang Huang, *Student Member, IEEE*, and Steve Renals, *Member, IEEE*

***Abstract*—Traditional n-gram language models are widely used in state-of-the-art large vocabulary speech recognition systems. This simple model suffers from some limitations, such as overfitting of maximum-likelihood estimation and the lack of rich contextual knowledge sources. In this paper, we exploit a hierarchical Bayesian interpretation for language modeling, based on a nonparametric prior called Pitman–Yor process. This offers a principled approach to language model smoothing, embedding the power-law distribution for natural language. Experiments on the recognition of conversational speech in multiparty meetings demonstrate that by using hierarchical Bayesian language models, we are able to achieve significant reductions in perplexity and word error rate.**

***Index Terms*—AMI corpus, conversational speech recognition, hierarchical Bayesian model, language model (LM), meetings, smoothing.**

I. Introduction

A LANGUAGE model (LM), which provides a predictive probability distribution for the next word based on a history of previously observed words, is an essential component of automatic speech recognition (ASR) systems. The dominant LM for most state-of-the-art large vocabulary ASR systems is the conventional n-gram model, which approximates the history as the immediately preceding $n-1$ words. Due to data sparsity, n-gram models based on maximum-likelihood estimation (MLE) severely overfit the training data, resulting in unseen events being assigned zero probabilities [1]. Assigning zero probabilities to events observed in a test set which were not observed in the training set is problematic for applications such as speech recognition and machine translation. The zero probability problem is addressed by *smoothing*, for which a large number of methods have been proposed in the literature [2], [3], including Good–Turing [4], Katz back-off [5], deleted interpolation [6], interpolated Kneser–Ney [7], and modified Kneser–Ney [2].

Although the n-gram LM has been demonstrated to be a simple but effective model, the struggle to improve over it continues. Broadly speaking, such attempts focus on the improved modeling of word sequences, or on the incorporation of richer knowledge. Approaches which aim to improve on maximum-likelihood n-gram models of word sequences include neural network-based models [8], latent variable models [9], and a Bayesian framework [10]–[12]. The exploitation of richer knowledge has included the use of morphological information in factored LMs [13], syntactic knowledge using structured LMs [14], and semantic knowledge such as topic information using nonparametric hierarchical Bayesian models [15].

In this paper, we propose the use of hierarchical Bayesian approaches [16] to better model of word sequences in language models, in the context of a practical large-vocabulary conversational speech recognition system. Bayesian models have explicitly declared prior assumptions, and an internally coherent framework for inference. They also have the advantages of incorporating additional knowledge sources and including themselves in larger models in a principled manner. More specifically, we present the application of hierarchical Pitman–Yor process language models (HPYLM) [12] on a large vocabulary meeting transcription system [17], using large training corpora. The HPYLM provides an alternative interpretation to language models in theory [11], [12], and a better smoothing algorithm for language modeling in practice [18].

The main goal of this paper is to carry out a comprehensive study on the application of the HPYLM to large vocabulary ASR. The HPYLM is a theoretically elegant language model first proposed in the machine learning field [12]. In the speech community, however, two questions remain interesting and have not been studied before. First, will the HPYLM work for ASR tasks, which are normally evaluated in terms of word error rate (WER)? Second, is it possible to scale up the HPYLMs to work on large-vocabulary ASR using large training corpora? In the rest of this paper, we provide our answers to these two questions, by extending our previous work in [18] including the presentation of a parallel training algorithm, more detailed descriptions, more experimental results, and a thorough discussion.

For the first question, we verify the HPYLM in terms of both perplexity and WER using an efficient computational implementation. In recent years there has been a growing research interest in the automatic transcription of multiparty meetings, which is typically one of the first several essential steps for further processing of meetings, such as information retrieval and summarization. European projects AMI and AMIDA [17] are

Manuscript received May 08, 2009; revised November 17, 2009. Date of publication January 19, 2010; date of current version September 01, 2010. This work was supported in part by the Wolfson Microelectronics Scholarship and in part by the European IST Program Projects FP6-506811 (AMI) and FP6-033812 (AMIDA). This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF) (http://www.ecdf.ed.ac.uk/), which is supported in part by the eDIKT initiative (http://www.edikt.org.uk/). This paper only reflects the authors' views and funding agencies are not liable for any use that may be made of the information contained herein. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Haizhou Li.

The authors are with the Centre for Speech Technology Research, University of Edinburgh, Edinburgh EH8 9AB, U.K. (e-mail: s.f.huang@ed.ac.uk; s.renals@ed.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2010.2040782

examples of such efforts. This has provided us a well-defined benchmark on which to evaluate a state-of-the-art large vocabulary ASR system. We present comprehensive experimental results on multiparty conversational meeting corpora, and observe consistent and significant reductions in perplexity and WER in comparison to the interpolated Kneser–Ney language model (IKNLM) [7] and the modified Kneser–Ney language model (MKNLM) [2], which are the state-of-the-art smoothing methods for language modeling.

It is often expensive to do Bayesian inference on large training data. In order to obtain a sufficiently large language model for state-of-the-art large-vocabulary ASR systems, we have developed a parallel algorithm for the estimation of an HPYLM, enabling the use of a large training corpus. We also demonstrate that inference of an HPYLM converges quickly, taking only a few tens of iterations to converge to a language model of comparable (or better) accuracy than the IKNLM or the MKNLM.

II. Hierarchical Bayesian Models

Bayesian analysis explicitly uses probability to quantify degrees of belief. Bayes' theorem (1) explains the relationship between the *prior*, the *likelihood*, and the *posterior*, where θ denotes the unknown parameters from the sample space Θ, and $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$ denotes the data from some sample space $\mathcal{X}$ (continuous or discrete)

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})} = \frac{P(\mathcal{D}|\theta)P(\theta)}{\int_{\Theta} P(\mathcal{D}|\theta)P(\theta)d\theta}. \quad (1)$$

The prior $P(\theta)$ represents the belief in the parameters θ before observing the data $\mathcal{D}$, and the posterior $P(\theta|\mathcal{D})$ represents the updated belief in the parameters after having observed the data $\mathcal{D}$. The likelihood is typically described using an exponential family distribution, which is mathematically convenient since conjugate priors exist for exponential family likelihood functions.[1] Bayesian modeling differs from MAP (maximum *a posteriori*) by computing the posterior distribution of θ rather than its maximum: this can be computationally challenging, due to the complexity of integrating over a large sample space Θ.

Hierarchical Bayesian models have been well studied and enable the construction of richer statistical models in which the prior may depend on parameters that are not involved in the likelihood [16]. The prior distribution $P(\theta)$ is itself a member of a family of densities with *hyperparameters* λ, $P(\theta|\lambda)$. The hierarchical Bayesian framework provides a better modeling of multi-parameter problems for hierarchical data, where non-hierarchical models usually tend to inappropriately overfit such data. Hierarchical Bayesian models can have enough parameters to fit the data well, while using a population distribution to structure some dependence into the parameters, thereby avoiding problems of overfitting [16].

A. Bayesian Priors

Bayesian analysis begins with a prior distribution capturing any available prior knowledge about the process generating the data. We will introduce several popular prior distributions in Bayesian data analysis, including Dirichlet distributions, Dirichlet processes, and Pitman–Yor processes.

1) Dirichlet Distribution: The Dirichlet distribution [19], a multivariate generalization of the beta distribution, is a density over a $(K-1)$-simplex, i.e., K-dimensional vectors $\mathbf{p}$ whose components p_i are all non-negative and sum to 1. The Dirichlet distribution is parameterized by a K-dimensional measure $\alpha\mathbf{m}$ where $\mathbf{m}$ is a normalized measure over K components $(\sum_i m_i = 1)$ and α is a positive scalar. The Dirichlet distribution with parameters $\alpha\mathbf{m}$ has a probability density function given by

$$P(\mathbf{p}|\alpha\mathbf{m}) = \frac{1}{Z(\alpha\mathbf{m})}\prod_{i=1}^{K} p_i^{\alpha m_i - 1} \quad (2)$$

with normalization constant $Z(\alpha\mathbf{m}) = \prod_{i=1}^{K}\Gamma(\alpha m_i)/\Gamma(\alpha)$ where $\Gamma()$ is the gamma function. The vector $\mathbf{m}$ represents the mean of the Dirichlet distribution, and α is a concentration parameter with larger values of α drawing samples away from the corners of the simplex to the centre, peaking around the mean $\mathbf{m}$. One reason why the Dirichlet distribution is selected as a prior is because it is conjugate to the multinomial distribution: i.e., the posterior is also a Dirichlet distribution when the prior is a Dirichlet distribution and the likelihood is a multinomial. We use $\mathrm{Dir}(\alpha\mathbf{m})$ to denote a Dirichlet density with hyperparameters $\alpha\mathbf{m}$. When $K = 2$, the Dirichlet distribution is equivalent to the beta distribution.

2) Dirichlet Process: The Dirichlet process (DP) is a stochastic process, first formalized in [20] for general Bayesian modeling, which has become an important prior distribution for nonparametric models. Nonparametric models are characterized by allowing the number of model parameters to grow with the amount of training data. This helps to alleviate over- or under-fitting problems, and provides an alternative approach to parametric model selection or averaging.

A random distribution G over a space Θ is called a Dirichlet process distributed with base distribution H and strength or concentration parameter α, if

$$(G(A_1), \ldots, G(A_r)) \sim \mathrm{Dir}(\alpha H(A_1), \ldots, \alpha H(A_r)) \quad (3)$$

for every finite measurable partition $A_1, \ldots, A_r$ of Θ [20]. We write this as $G \sim \mathrm{DP}(\alpha, H)$, and it may be interpreted as a distribution over distributions. The parameter H, a measure over Θ, is intuitively the mean of the DP. The parameter α, on the other hand, can be regarded as an inverse variance of its mass around the mean H, with larger values of α for smaller variances. More importantly in infinite mixture models, α controls the expected number of mixture components in a direct manner, with a larger α implying a larger number of mixture components *a priori*.

Draws from an DP are composed as a weighted sum of point masses located at the previous draws $\theta_1, \ldots, \theta_n$. This leads to a constructive definition of the DP called the stick-breaking construction [21]. If we construct G as follows:

$$\begin{array}{ll} \beta_k \sim \mathrm{Beta}(1,\alpha) & \theta_k^* \sim H \\ \pi_k = \beta_k \prod_{l=1}^{k-1}(1-\beta_\ell) & G = \sum_{k=1}^{\infty}\pi_k \delta_{\theta_k^*} \end{array}. \quad (4)$$

[1] A prior distribution is conjugate to the likelihood functions if Bayes' theorem results in a posterior distribution from the same family as the prior.

Then $G \sim \mathrm{DP}(\alpha, H)$. θ_k^* is a unique value among $\theta_1, \ldots, \theta_n$, and $\delta_{\theta_k^*}$ denotes a point mass at θ_k^*. The construction of $\boldsymbol{\pi}$ can be understood as follows [22]. Starting with a stick of length 1, first break it at β_1, assign π_1 to be the length of stick just broken off. Then recursively break the other portion to obtain π_2, π_3 and so forth. The stick-breaking distribution over $\boldsymbol{\pi}$ satisfies $\sum_{k=1}^{\infty} \pi_k = 1$ with probability one. This definition is important for inference of a DP.

Given observed values of $\theta_1, \ldots, \theta_n$, the posterior distribution of G is again distributed according to another DP with updated hyperparameters, with the following form [22]:

$$G|\theta_1, \ldots, \theta_n \sim \mathrm{DP}\left(\alpha + n, \frac{\alpha}{\alpha+n} H + \frac{n}{\alpha+n} \frac{\sum_{i=1}^{n} \delta_{\theta_i}}{n}\right) \tag{5}$$

where the posterior base distribution is a weighted average between the prior base distribution H and the empirical distribution $\sum_{i=1}^{n} \delta_{\theta_i}/n$.

3) Pitman–Yor Process: The Pitman–Yor process or the two-parameter Poisson–Dirichlet process [23], [24] $\mathrm{PY}(d, \theta, G_b)$ is a three-parameter distribution over distributions, where d is a discount parameter, θ a strength parameter, and G_b a base distribution that can be understood as a mean of draws from $\mathrm{PY}(d,\ \theta,\ G_b)$. When $d = 0$, the Pitman–Yor process reverts to the Dirichlet process $\mathrm{DP}(\theta,\ G_b)$. In this sense, the Pitman–Yor process is a generalization of the Dirichlet process.

The procedure for generating draws $\mathrm{G} \sim \mathrm{PY}(d,\ \theta,\ G_b)$ from a Pitman–Yor process can be described using the "Chinese Restaurant" metaphor [24], [25]. Imagine a Chinese restaurant containing an infinite number of tables, each with infinite seating capacity. Customers enter the restaurant and seat themselves. The first customer sits at the first available table, while each of the subsequent customers sits at an occupied table with probability proportional to the number of customers already sitting there $c_k - d$, or at a new unoccupied table with probability proportional to $\theta + dt_\bullet$, where $t_\bullet$ is the current number of occupied tables. That is, if z_i is the index of the table chosen by the ith customer, then the ith customer sits at table k given the seating arrangement of the previous $i-1$ customers $\mathbf{z}_{-i} = \{z_1, \ldots, z_{i-1}\}$ with probability

$$P(z_i = k|\mathbf{z}_{-i}) = \begin{cases} \frac{c_k - d}{\theta + c_\bullet}, & 1 \le k \le t_\bullet \\ \frac{\theta + dt_\bullet}{\theta + c_\bullet}, & k = t_\bullet + 1 \end{cases} \tag{6}$$

where c_k is the number of customers sitting at table k and $c_\bullet = \sum_k c_k$ is the total number of customers. The Pitman–Yor process with parameters (d, θ, G_b) produces a power-law distribution with index $1 + d$ over the number of customers seated at each table [11]. The power-law distribution—a few outcomes have very high probability and most outcomes occur with low probability—has been found to be one of the most striking statistical properties of word frequencies in natural language.

B. Bayesian Inference

In general, there are two types of approximate inference algorithms in Bayesian analysis for the estimation of posterior distributions of interest: Monte Carlo methods and variational methods. Monte Carlo methods [16], [26] use random samples to simulate probabilistic models. They are guaranteed to give arbitrarily precise estimates with sufficient computation. *Markov chain Monte Carlo (MCMC)* methods are a family of iterative Monte Carlo algorithms that draw samples from an otherwise intractable target density via a first-order Markov process. Gibbs sampling, a special case of the Metropolis–Hastings algorithm [26], is one of the most widely used MCMC methods for Bayesian inference. It assumes that it is tractable to sample from the conditional distribution of one of these variables given the other $(N-1)$ ones. We will use Gibbs sampling methods for inference in this paper.

Variational methods [27], [28], on the other hand, are a class of deterministic approximations to the problems of learning and inference for Bayesian inference. A variational method begins by expressing a statistical inference task as the solution to a mathematical optimization problem [29]. By approximating or relaxing the objective function, one can derive computationally tractable algorithms which bound or approximate the statistics of interest. Sudderth [29] gives a good review and comparison of various Bayesian inference algorithms.

III. Smoothing N-Gram Language Models

The goal of an n-gram model is to estimate the conditional probability distribution over next words given the context $\mathbf{u}$ (approximated by the immediately preceding $n-1$ words) from a training corpus:

$$P(w_n|\mathbf{u}) \approx P(w_n|w_1, w_2, \ldots, w_{n-1}). \tag{7}$$

The simplest way to estimate the required n-gram probabilities is maximum-likelihood estimation, which uses the frequencies of co-occurrences of word w following the context $\mathbf{u}$, and maximizes the likelihood over the training data. This has a poor generalization ability because n-gram training data is sparse, leading to zero probability estimates for n-grams not observed in the training corpus. In order to alleviate this problem, a number of smoothing techniques have been proposed for n-gram models estimated using maximum likelihood. Good–Turing smoothing [4], [5], which is based on leave-one-out estimation, re-estimates the probability mass assigned to n-grams with zero counts by making use of the frequency of n-grams occurring only once. Absolute discounting [30], the root of Kneser–Ney smoothing [7], subtracts a fixed absolute discount d from each nonzero count, and redistributes the unallocated probability mass to those unseen events. In general, it is also useful to take advantage of the n-gram hierarchy for smoothing methods through back-off [5] or interpolation [6]. Back-off relies on only the lower order n-grams when smoothing unseen events, while interpolation linearly interpolates higher order n-gram models with lower order n-gram models. To obtain a smoother LM, interpolated Kneser–Ney smoothing [7] utilizes absolute discounting, modified counts for lower order m-gram probabilities $(m < n)$, and interpolation with low order m-gram probabilities

$$\begin{aligned} P(w|\mathbf{u}) &\approx P_{\mathbf{u}}^{\mathrm{IKN}}(w) \\ &= \frac{\max(c_{\mathbf{u}w} - d_{|\mathbf{u}|}, 0)}{c_{\mathbf{u}\bullet}} + \frac{d_{|\mathbf{u}|} t_{\mathbf{u}\bullet}}{c_{\mathbf{u}\bullet}} P_{\pi(\mathbf{u})}^{\mathrm{IKN}}(w) \end{aligned} \tag{8}$$

where $c_{\mathbf{u}\bullet} = \sum_{w'} c_{\mathbf{u}w'}$ is the total number of word tokens following the context $\mathbf{u}$, $\pi(\mathbf{u})$ is the context one word shorter than $\mathbf{u}$, $t_{\mathbf{u}\bullet} = |\{w' : c_{\mathbf{u}w'} > 0\}|$ is the number of distinct words w' occurring after $\mathbf{u}$, and the discount $d_{|\mathbf{u}|}$ is dependent on the length of the context. Modified Kneser–Ney smoothing extends interpolated Kneser–Ney smoothing by allowing three different discount parameters, $d_{|\mathbf{u}|1}$, $d_{|\mathbf{u}|2}$, and $d_{|\mathbf{u}|3+}$ for n-grams with one, two, and three or more counts, respectively [2]. Counts of counts statistics are used to estimate the optimal values for average discounts in the MKNLM. As an alternative to these smoothing schemes, class-based n-gram language models [31] have been used, in which word classes or clusters (either hand-designed or automatically induced) help to address the data sparsity problem.

In the Bayesian framework for language modeling, a prior distribution is placed over the predictive distribution of interest for LMs in (7), and the posterior distribution is inferred from the observed training data. The final predictive probability can then be estimated from the posterior by marginalizing out the latent variables and hyperparameters. The Bayesian interpretation essentially avoids the zero-probability problem to estimate smoother LMs by taking advantage of knowledge expressed by the priors.

There has been considerable prior work in which prior distributions are placed over LM parameters. Nádas [32] used a beta-binomial model (the beta distribution is the conjugate prior to the binomial) in which the so-called "empirical Bayes" method was used to obtain point estimates of the hyperparameters of the prior distribution, by maximizing the likelihood on the training data rather than by full Bayesian inference. More recently Yaman *et al.* [33] proposed a structural Bayesian language modeling and adaptation framework, employing a Dirichlet prior density on n-grams assembled in a tree structure. In this case MAP estimation was employed to estimate the language model parameters.

Goodman [34] analyzed back-off Kneser–Ney smoothing in a Bayesian manner, in which an exponential prior distribution is used in conjunction with a maximum entropy model. This analysis provides a justification for Kneser–Ney smoothing, in terms of the discounting procedure and the satisfaction of marginal constraints when estimating lower order n-grams.

A full Bayesian approach to language modeling was introduced by MacKay and Peto [10], which extended Nádas' empirical Bayes framework to a hierarchical Dirichlet LM, by using Dirichlet distributions as the priors. The predictions of hierarchical Dirichlet LMs are similar to those of a traditionally smoothed LM. MacKay and Peto demonstrated in this way, on a small corpus, that a hierarchical Dirichlet language model had comparable performance to a bigram model smoothed by deleted interpolation with specific values of interpolation weight.

It was argued by Goldwater *et al.* [11] that a Pitman–Yor process is more suitable as a prior distribution than a Dirichlet distribution to applications in natural language processing, as the power-law distributions of word frequencies produced by Pitman–Yor processes more closely resemble the heavy-tailed distributions observed in natural language. The hierarchical extension of the Pitman–Yor process—HPYLM—was independently proposed for language modelling by Goldwater *et al.* [11] and by Teh [12]. The HPYLM can be considered as a natural generalization of the hierarchical Dirichlet language model [10], by using a Pitman–Yor process rather than the Dirichlet distribution. Experiments on the AP News corpus showed that the novel hierarchical Pitman–Yor process language model produces results superior to hierarchical Dirichlet language models and n-gram LMs smoothed by interpolated Kneser–Ney (IKN), and comparable to those smoothed by modified Kneser–Ney (MKN) [12]. Wood and Teh [35] additionally extended the hierarchical Hierarchical Bayesian Language Models for Conversational Speech Recognition language model for domain adaptation of language models.

IV. Hierarchical Bayesian Language Models Based on Pitman–Yor Processes

We introduce a Bayesian language model based on Pitman–Yor processes using a hierarchical framework. This section briefly summarizes the original work on the HPYLM [11], [12], and refers to our previous work [18].

A. Hierarchical Pitman–Yor Process Language Models

The Pitman–Yor process can be used to create a two-stage language modeling framework [11]. Following the Chinese restaurant metaphor discussed in Section II-A3, a language model can be viewed as a restaurant in which each table has a label of a word w generated by $G_b(w)$. Each customer represents a word token, so that the number of customers at a table corresponds to the frequency of the lexical word labeling that table. A customer may only be assigned to a table whose label matches that word token.

Consider a vocabulary $\mathcal{V}$ with $|\mathcal{V}|$ word types. Let $G_\emptyset(w)$ be the unigram probability of w, and $G_\emptyset = [G_\emptyset(w)]_{w\in\mathcal{V}} = [G_\emptyset(w_1), G_\emptyset(w_2), G_\emptyset(w_3), \ldots, G_\emptyset(w_{|\mathcal{V}|})]$ represents the vector of word probability estimates for unigrams. A Pitman–Yor process prior is placed over $G_\emptyset \sim \mathrm{PY}(d_0, \theta_0, G_b)$ with an uninformative base distribution $G_b(w) = 1/|\mathcal{V}|$ for all $w \in \mathcal{V}$. According to the Chinese restaurant metaphor, customers (word tokens) enter the restaurant and seat themselves at either an occupied table or a new one, with probabilities expressed in (6). Each table has a label $w \in \mathcal{V}$ initialized by the first customer seated on it, and the next customer can only sit on those tables with the same label. Those c_w customers that correspond to the same word label w, can sit at different tables, with t_w denoting the number of tables with labels w. Given the seating arrangement $\mathcal{S}$ of customers, and the hyperparameters d_0 and θ_0, the predictive probability of a new word w is given in (9), by collecting probabilities in (6) corresponding to each label w for tables

$$\begin{aligned} P(w|\mathcal{S}, d_0, \theta_0) &= \sum_{k=1}^{t_\bullet} \frac{c_k - d_0}{\theta_0 + c_\bullet}\delta_{kw} + \frac{\theta_0 + d_0 t_\bullet}{\theta_0 + c_\bullet} G_b(w) \\ &= \frac{c_w - d_0 t_w}{\theta_0 + c_\bullet} + \frac{\theta_0 + d_0 t_\bullet}{\theta_0 + c_\bullet} G_b(w) \end{aligned} \tag{9}$$

where δ_{kw} equals to 1 if table k has the label of w, and 0 otherwise, $c_\bullet = \sum_w c_w$ is the total number of customers, and $t_\bullet = \sum_w t_w$ is the total number of tables, in the restaurant for

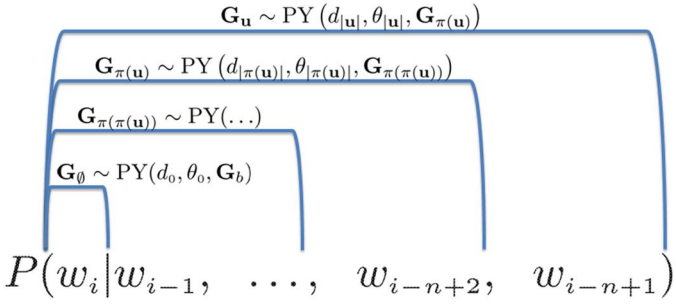


Fig. 1. Hierarchy of Pitman–Yor process priors for n-gram LMs. Pitman–Yor processes are placed recursively as priors over the n-order predictive distributions until we reach the unigram model $G_{\emptyset}$. $\pi(\mathbf{u})$ denotes the back-off context of $\mathbf{u}$.

unigrams. Averaging over seating arrangements and hyperparameters, we can obtain the probability $P(w)$ for a unigram LM. When $d_0 = 0$, the Pitman–Yor process reduces to a Dirichlet distribution $\mathrm{Dir}(\theta_0 G_b)$, and (9) becomes the predictive distribution in a Bayesian language model using the Dirichlet distribution as the prior [10]

$$P(w|\mathcal{S}, \theta_0) = \frac{c_w}{\theta_0 + c_\bullet} + \frac{\theta_0 G_b(w)}{\theta_0 + c_\bullet} = \frac{c_w + \theta_0 G_b(w)}{c_\bullet + \theta_0}. \tag{10}$$

This can be regarded as an additive smoothing of the empirical probability $(c_w/c_\bullet)$, by balancing the empirical counts (c_w) with the additive pseudo-counts $(\theta_0 G_b(w))$ of the prior Dirichlet distribution.

We can generalize the above unigram example to the n-gram case. An n-gram LM defines a probability distribution over the current word given a context $\mathbf{u}$ consisting of $n-1$ words. Let $G_{\mathbf{u}}(w)$ be the probability of the current word w and $G_{\mathbf{u}} = [G_{\mathbf{u}}(w)]_{w \in \mathcal{V}}$ be the target probability distribution given the context $\mathbf{u}$. A Pitman–Yor process is served as the prior over $G_{\mathbf{u}}$, with discounting parameter $d_{|\mathbf{u}|}$ and strength parameter $\theta_{|\mathbf{u}|}$ specific to the length of the context $|\mathbf{u}|$. The base distribution is $G_{\pi(\mathbf{u})}$, the lower order model of probabilities of the current word given all but the earliest word in the context. That is,

$$G_{\mathbf{u}} \sim \mathrm{PY}\left(d_{|\mathbf{u}|}, \theta_{|\mathbf{u}|}, G_{\pi(\mathbf{u})}\right). \tag{11}$$

Since $G_{\pi(\mathbf{u})}$ is still an unknown probability distribution, a Pitman–Yor process is recursively placed over it with parameters specific to $|\pi(\mathbf{u})|$, $G_{\pi(\mathbf{u})} \sim \mathrm{PY}(d_{|\pi(\mathbf{u})|}, \theta_{|\pi(\mathbf{u})|}, G_{\pi(\pi(\mathbf{u}))})$. This is repeated until we reach $G_{\emptyset}$ for a unigram model discussed above. This results in a hierarchical prior (Fig. 1), enabling us to generalize from the unigram to the n-gram case. There are multiple restaurants (Pitman–Yor processes) in the prior hierarchy, with each corresponding to one context. By using the hierarchical framework of Pitman–Yor priors, different orders of n-gram can thus share information with each other, similar to the traditional interpolation of higher order n-grams with lower order n-grams.

Based on this overall framework for an HPYLM, a central task is the inference of seating arrangements in each restaurant and the estimation of the context-specific parameters from the training data. Given training data $\mathcal{D}$, we know the number of co-occurrences of a word w after a context $\mathbf{u}$ of length $n-1$, $c_{\mathbf{u}w}$. This is the only information we need to train an HPYLM. An MCMC algorithm can be used to infer the posterior distribution of seating arrangements. We use Gibbs sampling to keep track of which table each customer sits at, by iterating over all customers present in each restaurant—first removing a customer w from the restaurant $\mathbf{u}$, and then adding the customer w back to the restaurant $\mathbf{u}$ by resampling the table at which that customer sits. After a sufficient number of iterations, the states of variables of interest in the seating arrangements will converge to the required samples from the posterior distribution. In the HPYLM the more frequent a word token, the more likely it is there are more tables corresponding to that word token.

For an n-gram LM, there are $2n$ parameters $\Theta = \{d_m, \theta_m : 0 \leq m \leq n-1\}$ to be estimated in total. We use a sampling method based on auxiliary variables [36].

Under a particular setting of seating arrangements $\mathcal{S}$ and hyperparameters Θ, the predictive probability $P(w|\mathbf{u}, \mathcal{S}, \Theta)$ can be obtained similarly to the case for unigram in (9) for each context $\mathbf{u}$

$$P(w|\mathbf{u}, \mathcal{S}, \Theta) = \frac{c_{\mathbf{u}w\bullet} - d_{|\mathbf{u}|} t_{\mathbf{u}w}}{\theta_{|\mathbf{u}|} + c_{\mathbf{u}\bullet\bullet}} + \frac{\theta_{|\mathbf{u}|} + d_{|\mathbf{u}|} t_{\mathbf{u}\bullet}}{\theta_{|\mathbf{u}|} + c_{\mathbf{u}\bullet\bullet}} P(w|\pi(\mathbf{u}), \mathcal{S}, \Theta) \tag{12}$$

in which if we set the discounting parameters $d_{|\mathbf{u}|} = 0$ for all $\mathbf{u}$, we resort to a hierarchical Dirichlet language model (HDLM) [10], similar to (10). The HDLM and the HPYLM share the same idea of interpolation with the lower order n-grams. The difference is that the HPYLM explores discounts from empirical counts, while the HDLM does not.

The overall predictive probability can be approximately obtained by collecting I samples from the posterior over $\mathcal{S}$ and Θ, and then averaging (12) to approximate the integral with samples

$$P(w|\mathbf{u}) \approx \sum_{i=1}^{I} \frac{P(w|\mathbf{u}, \mathcal{S}^{(i)}, \Theta^{(i)})}{I}. \tag{13}$$

If we assume that the strength parameters $\theta_{|\mathbf{u}|} = 0$ for all $\mathbf{u}$, and restrict $t_{\mathbf{u}w}$ to be at most 1 (i.e., all customers representing the same word token should only sit on the same table together), then the predictive probability in (12) directly reduces to the predictive probability given by the IKNLM in (8). We can thus interpret IKN as an approximate inference scheme for the hierarchical Pitman–Yor process language model [12].

B. Inference

In the HPYLM, we are interested in the posterior distribution over the latent predictive probabilities $\mathcal{G} = \{G_{\mathbf{u}} : \text{all contexts } \mathbf{u}\}$ and the hyperparameters $\Theta = \{d_m, \theta_m : 0 \leq m \leq n-1\}$, given the training data $\mathcal{D}$. The hierarchical Chinese restaurant process represents it as the seating arrangement, denoted by $\mathcal{S} = \{S_u : \text{all contexts } \mathbf{u}\}$, in the corresponding restaurant, by marginalizing out each $G_{\mathbf{u}}$. The central task for the inference is thus to infer the posterior distribution over the seating arrangements $\mathcal{S} = \{S_u : \text{all contexts } \mathbf{u}\}$ and the

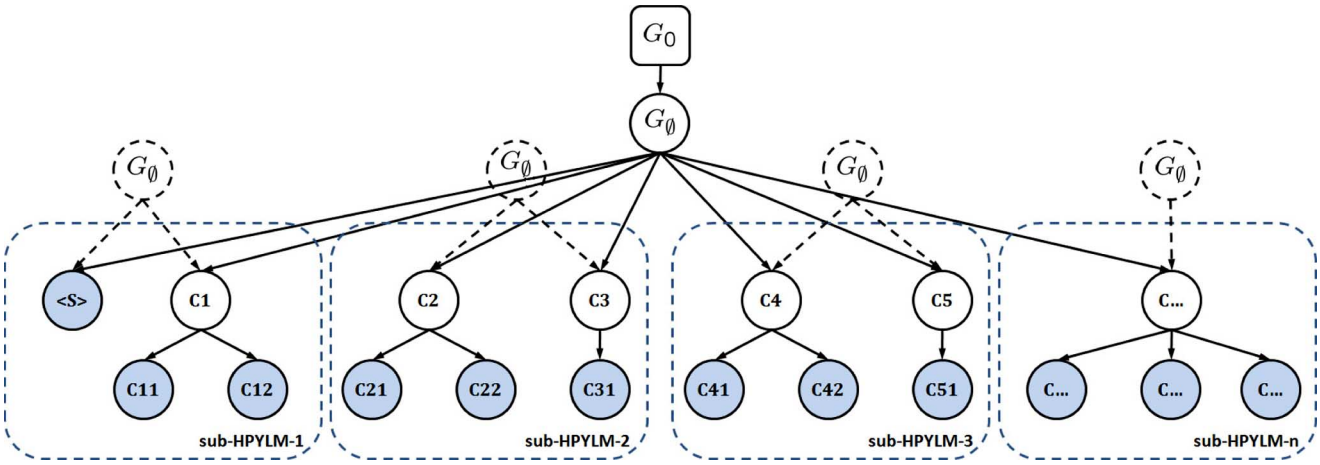


Fig. 2. Partition of an HPYLM into sub-HPYLMs, denoted by dotted rectangles, for the parallel inference. The dotted circles represent pseudo $G_\emptyset$s to complete a Pitman–Yor process hierarchy, and collect additional insertion/deletion information. Each circle corresponds to a context, or a restaurant in the Chinese restaurant metaphor.

hyperparameters $\Theta = \{d_m,\ \theta_m\ :\ 0 \leq m \leq n-1\}$ given the training data $\mathcal{D}$ $P(\mathcal{S}, \Theta|\mathcal{D}) = P(\mathcal{S},\ \Theta,\ \mathcal{D})/P(\mathcal{D})$. With the posterior, we can calculate the predictive probabilities according to (12) and (13) by further integrating out $\mathcal{S}$ and Θ. We follow inference schemes based on MCMC for the HPYLM [12], [36], depicted in algorithm 1.

Algorithm 1 InferHpylm (n): An Algorithm for the Inference of an HPYLM, Where n is the Order of LMs, $\mathbf{u}$ is the Context (Restaurant), w is the Word Token (Customer), n_w is the Number of Occurrences of w After Context $\mathbf{u}$. d_m, and θ_m are the Discount and Strength Parameters of Pitman–Yor Processes for n-Grams of Order m.

procedure INFERHPYLM (order n)

input: the order of n-gram with $n > 0$

```
for each order m = 0 to n − 1 do
  for each context u of order m do
    for each word w appearing after context u do
      for i = 0 to n_w do iterations for n_w times
        if REMOVECUSTOMER (u, w) then
          ADDCUSTOMER (u, w);
        endif
      endfor
    endfor
  endfor
  (d_m, θ_m) = SAMPLEPARAMS (m); /* sampling
  hyperparameters d_m, θ_m for order m */
endfor
```

For ADDCUSTOMER $(\mathbf{u}, w)$, we use (6) to sample a new seating table in restaurant $\mathbf{u}$ for customer w. For REMOVECUSTOMER $(\mathbf{u}, w)$, we simply remove a customer from k^{th} table according to the probability proportional to the number of customers already seated there $c_{\mathbf{u}wk}$. For SAMPLEPARAMS (m), a sampling method based on auxiliary variables is used for sampling the hyperparameters d_m and θ_m, which assumes that each discount parameter d_m has a prior beta distribution $d_m \sim \text{Beta}(a_m, b_m)$ while each strength parameter θ_m has a prior gamma distribution $\theta_m \sim \text{Gamma}(\alpha_m, \beta_m)$ [12], [36].

C. Parallelization

It is computationally expensive in terms of computing time and memory requirements to infer an HPYLM from a large corpus. This motivated us to design a divide-and-conquer algorithm to efficiently estimate an HPYLM. The algorithm has two steps: *data partition* and *model combination*. Generally speaking, we divide the inference task, which is normally infeasible or expensive using a single machine, into sub-tasks that fit well to the computational capacity of a single computing node—alleviating the memory requirement and allowing sub-tasks to be run in parallel.

In the data partition step, we first divide word types in the vocabulary $\mathcal{V}$ into subsets $\mathcal{V}_k \subset \mathcal{V}$. For each subset $\mathcal{V}_k$, we then compose those bigrams beginning with words $w \in \mathcal{V}_k$, and their corresponding child n-grams with $n > 2$, as a sub-HPYLM (dotted rectangles), and put a pseudo $G_\emptyset$ (dotted circles) as the Pitman–Yor process for unigrams of the sub-HPYLM, as shown in Fig. 2. Each sub-HPYLM can be inferred separately using the same routines as those for a normal HPYLM, except that the pseudo $G_\emptyset$ now additionally collects the number of insertion and deletion for customer $w \in \mathcal{V}_k$. The inference of sub-HPYLMs can be executed in parallel.

In the model combination step, we combine all the sub-HPYLM models level-by-level in the HPY hierarchy. For each level, we accumulate auxiliary parameters, and sample the hyperparameters d and θ. For the global $G_\emptyset$ for unigrams, we infer the seating arrangements by using the insertion and deletion statistics accumulated by each pseudo $G_\emptyset$,

TABLE I
STATISTICS OF THE TRAINING AND TESTING DATA SETS FOR LANGUAGE MODELS IN THE RT06S TASK

No.	LM Data Set	#Sentences	#Words
1	AMI data from rt05s	68,806	801,710
2	ICSI meeting corpus	79,307	650,171
3	ISL meeting corpus	17,854	119,093
4	NIST meeting corpus-2	21,840	156,238
5	NIST meeting corpus-a	18,007	119,989
	MTRAIN (Sets 1–5)	205,814	1,847,201
6	FISHER (fisher-03-p1)	1,076,063	10,593,403
	MTRAIN + F (Sets 1–6)	1,281,877	12,440,604
7	WEBMEET (webdata meetings)	3,218,066	36,073,718
	ALL - WC (Sets 1–7)	4,499,943	48,514,322
8	WEBCONV (webdata conversational)	12,684,025	162,913,566
	ALL	17,183,968	211,427,888
test	*rt06seval*	3,597	31,810

to make sure the HPYLM is consistent regarding the *modified* counts for lower order n-grams [7], [12].

One approximation we made during this parallelism is the inference for unigram $G_\emptyset$, which is inferred based on pseudo $G_\emptyset$ of sub-HPYLMs and not globally optimized. However, in Section VI-C we show experimentally that this approximation does not significantly affect the performance in terms of perplexity or WER.

The parallel training algorithm enables us to estimate an HPYLM on a large corpus using a large vocabulary. We describe the parallel training algorithm for the HPYLM in more detail in [37].

D. Implementation

We implemented the hierarchical Pitman–Yor process language model by extending the SRILM toolkit [38]. We highlight four characteristics of this implementation. First, it is consistent and coherent with the existing SRILM software. We inherited the HPYLM classes from the base SRILM classes, and provided the same interfaces for language modeling. Second, it has efficient memory management and computational performance by directly using the data structures available in SRILM. Third, it is a flexible framework for Bayesian language modeling. We can, for example, train a language model with Kneser–Ney smoothing for unigrams, modified Kneser–Ney smoothing for bigrams, and Pitman–Yor process smoothing for trigrams. Finally, this implementation is extensible for future developments: e.g., taking into accounts the combination with multimodal cues for language models via probabilistic topic models.

This implementation of an HPYLM outputs a standard ARPA format LM, with an identical format to a conventional n-gram LM. This makes it easy to evaluate the HPYLM in a conventional ASR system.

V. EXPERIMENTS AND RESULTS

A. Experimental Setup

The experiments reported in this paper were performed using the U.S. National Institute of Standard and Technology (NIST) Rich Transcription (RT) 2006 spring meeting recognition evaluation data (RT06s).[2] We tested only on those audio data recorded from individual headset microphones (IHM), consisting of meeting data collected by the AMI project, Carnegie Mellon University (CMU), NIST, and Virginia Tech (VT).

TABLE II
STATISTICS OF THE FIVEFOLD CROSS VALIDATION SETUP OF THE AMI SCENARIO MEETINGS

FOLD	#Sentences	#Words	#Meetings
1	17,368	175,302	32
2	13,512	135,415	28
3	12,220	129,212	24
4	12,849	136,071	28
5	11,975	113,969	25
all	67,924	689,969	137

The training data sets for language models used in this paper, and the test transcription, are listed in Table I. The web-data for meetings and conversational speech were collected from the world wide web using strategies described in [39]. We exploited different combinations of these training data sets for the following experiments.

A second corpus we consider in this paper is a domain-specific meeting corpus—the AMI Meeting Corpus,[3] [40] which consists of 100 hours of multimodal meeting recordings with comprehensive annotations at a number of different levels. About 70% of the corpus was elicited using a design scenario, in which the participants play the roles of employees—project manager, marketing expert, user interface designer, and industrial designer—in an electronics company that decides to develop a new type of television remote control. We used the scenario part of the AMI Meeting Corpus for our experiments, in a fivefold cross validation setup. There are 137 scenario meetings in total, as shown in Table II.

Most of the following experiments used a common vocabulary with 50 000 word types, unless explicitly indicated otherwise. For the fivefold cross-validation experiments on the scenario AMI meetings, the vocabulary was slightly tuned for each fold, while keeping the vocabulary size fixed to 50 000 word types. These vocabularies were those used in the AMI-ASR system [41].

The lower discounting cutoffs of the n-gram counts (i.e., $-gt1min$, $-gt2min$, and $-gt3min$ for ngram-count in the SRILM toolkit [38]) were set to 1 in all the LMs used in the following experiments.

B. Perplexity Experiments

We took the LM data sets from No.1 to No.5 in Table I as a core training set, named MTRAIN, which consists of 205 814 sentences and 1 847 201 words. We trained trigram IKN, MKN, HD, and HPY LMs using this training data. For the HDLM and the HPYLM, we ran 200 iterations for inference, and collected 100 samples from the posterior over seating arrangements and hyperparameters.

[2]http://www.nist.gov/speech/tests/rt/

[3]http://corpus.amiproject.org

TABLE III
PERPLEXITY RESULTS ON *RT*06SEVAL TESTING DATA

Condition	VOCAB	UNK	IKNLM	MKNLM	HDLM	HPYLM
OV	open	no	95.7	93.5	100.1	**88.6**
OV+U	open	yes	122.0	119.2	139.9	**111.9**
CV	close	no	110.1	106.5	118.5	**101.2**
CV+U	close	yes	110.5	106.8	120.4	**102.6**

TABLE IV
PERPLEXITY RESULTS OF THE FIVEFOLD CROSS VALIDATION ON THE AMI SCENARIO MEETINGS

FOLD	IKNLM	MKNLM	HDLM	HPYLM
1	107.1	104.5	116.6	**100.3**
2	99.3	97.1	107.8	**93.7**
3	106.5	103.9	115.8	**100.1**
4	101.8	99.5	110.3	**96.1**
5	101.1	98.5	108.2	**94.5**
average	103.2	100.7	111.7	**96.9**

The test data for perplexity estimation was extracted from the reference transcriptions for *rt06seval*. The final test data consisted of 3597 sentences and 31 810 words. Four different experimental conditions were considered and are shown in Table III: the combination of whether or not a closed vocabulary (CV/OV) was used and/or mapping unknown words to a special symbol "UNK" $(+\mathrm{U})$ during training and testing. If the "UNK" symbol is not included in the vocabulary, all unknown words will be skipped from the training data for the estimation and from the testing data for the evaluation.

Table III shows the perplexity results. We can see that in all four experiment conditions, the HPYLM has a lower perplexity than both the IKNLM and the MKNLM, and the HDLM has the highest perplexity, which is as expected and consistent with the previous results in [12]. We used the CV condition, i.e., with a closed vocabulary but without the "UNK" symbol, when training an LM in all the rest of experiments.

Table IV shows the perplexity results on the fivefold cross validation of the scenario AMI Meeting Corpus, using the CV condition. The HPYLM again has a consistently lower perplexity than both the IKNLM and the MKNLM, and the HDLM again is the worst.

C. Word Error Rate Experiments

We used the AMI-ASR system [41] as the baseline platform for our ASR experiments. The feature stream comprised of 12 MF-PLP features and raw log energy and first– and second-order derivatives are added. Cepstral mean and variance normalisation was performed on a per channel basis. The acoustic models were taken from the second pass of AMI-ASR system, which were trained on 108 hours speech data from ICSI, ISL, NIST, and AMI, using vocal tract length normalization, heteroscedastic linear discrimant analysis, speaker adaptive training, and minimum phone error discriminative training. They are adapted using the transcripts of the first pass and a single constrained maximum-likelihood linear regression transform. We only tested LMs trained using training data MTRAIN (see Table I) under condition CV in Table III, that is, we used a 50 k vocabulary but without mapping unknown words to "UNK" during training. For the HDLM and the HPYLM, we output an ARPA format LM. Different LMs were then used in the first pass decoding using `HDecode`.[4]

TABLE V
WER (%) RESULTS ON *RT*06SEVAL TESTING DATA

LMS	SUB	DEL	INS	WER
IKNLM	15.7	9.9	2.9	28.5
MKNLM	15.6	10.0	2.8	28.4
HDLM	16.4	10.4	3.0	29.8
HPYLM	**15.3**	**10.1**	**2.7**	**28.1**

Table V shows the WER results for the RT06s task. Unsurprisingly, the HDLM produces the highest WER. The HPYLM, however, results in a lower WER than both the IKNLM and the MKNLM. These reductions by the HPYLM from the IKNLM and the MKNLM are both significant using a matched-pair significance test [42], with $p < 0.0005$ and $p < 0.02$, respectively. This is an encouraging result, since it is the first time that the HPYLM has been tested using a state-of-the-art large-vocabulary ASR system on standard evaluation data.

Table VI shows the WER results for the fivefold cross-validation experiments on the scenario AMI meetings. We observed statistically significant $(p < 0.001)$ reductions in WER by the HPYLM, which are consistent among all the five folds and the scenario AMI meetings as a whole. Our experiments also show that the HDLM consistently gives highest WERs. We therefore stop presenting results for the HDLM in the rest of experiments. We used only the transcriptions of the scenario AMI meetings to train LMs.

D. Scalability

To investigate the scalability of the HPYLM, we gradually increased the size of training data for the HPYLM, as shown in Table VII. MTRAIN includes the training data sets No.1–5. $\mathrm{MTRAIN} + \mathrm{F}$ consists of MTRAIN and the No.6 data set Fisher-p1. Further adding the data set No.7 to $\mathrm{MTRAIN}+\mathrm{F}$ we obtained ALL-WC. Finally, we put together all the data No.1–8 as shown in Table I, named ALL.

For MTRAIN, $\mathrm{MTRAIN} + \mathrm{F}$, and ALL-WC, experiments were carried out on a machine with dual quad-core Intel Xeon 2.8-GHz processors and 12 GB of memory. Table VII shows the computational time per iteration and memory requirements when we change the size of training data, or vary the size of the vocabulary. From the results in Table VII, we can see that the training time for each iteration scales linearly with the size of training data when vocabulary size is constant. The smaller the size of the vocabulary, the quicker each iteration and the lower the memory requirement. The observations confirm the necessity of proposing a parallel training algorithm for the HPYLM. For IKNLM and MKNLM trained on ALL-WC, the memory requirement is around 1 GB.

For ALL, it would be extremely demanding to train an HPYLM on this data set using a single machine, due to the computational time and memory limitations. We instead used the parallel training algorithm described in Section IV-D. We

[4]http://htk.eng.cam.ac.uk/

TABLE VI
WER (%) RESULTS OF FIVEFOLD CROSS VALIDATION ON THE SCENARIO AMI MEETINGS. ALL THE REDUCTIONS BY THE HPYLM WITH RESPECT TO THE IKNLM AND THE MKNLM ARE STATISTICALLY SIGNIFICANT, WITH $p < 0.001$

FOLD	LMS	SUB	DEL	INS	WER
1	IKNLM	22.1	11.3	5.6	39.0
	MKNLM	21.9	11.4	5.5	38.9
	HPYLM	**21.7**	**11.5**	**5.4**	**38.6**
2	IKNLM	21.0	11.2	5.3	37.5
	MKNLM	20.8	11.4	5.2	37.4
	HPYLM	**20.7**	**11.6**	**5.0**	**37.3**
3	IKNLM	22.3	11.3	5.3	38.9
	MKNLM	22.2	11.4	5.1	38.7
	HPYLM	**21.9**	**11.5**	**4.9**	**38.3**
4	IKNLM	20.9	11.7	5.7	38.4
	MKNLM	20.9	11.3	5.6	37.8
	HPYLM	**20.7**	**11.3**	**5.5**	**37.5**
5	IKNLM	24.5	12.6	6.4	43.6
	MKNLM	24.2	13.4	6.1	43.7
	HPYLM	**24.1**	**12.9**	**6.1**	**43.1**
all	IKNLM	22.1	11.6	5.7	39.3
	MKNLM	21.9	11.7	5.5	39.1
	HPYLM	**21.7**	**11.7**	**5.3**	**38.8**

TABLE VII
COMPARISON OF COMPUTATIONAL TIME AND MEMORY REQUIREMENT OF THE HPYLM ON DIFFERENT TRAINING DATA SETS. DATA SET NUMBERS REFER TO TABLE I

Name	Data	#Words	Vocab	Time/Iter	Memory
MTRAIN	No.1–5	1,847,201	50k	∼10sec	∼150MB
MTRAIN+F	No.1–6	12,440,604	50k	∼120sec	∼600MB
ALL-WC	No.1–7	48,514,322	50k	∼600sec	∼2400MB
			18k	∼300sec	∼2000MB
			8k	∼200sec	∼1400MB
ALL	No.1–8	211,427,888	50k	parallel	parallel

divided the inference into 50 sub-tasks. It turns out that it is feasible to train an HPYLM on such a large corpus of more than 200 million word tokens, using a vocabulary of 50 k work types in this way. The inference took around one day to finish 100 iterations, although this was highly dependent on submission and queueing times of the compute cluster. For ALL, we evaluated two HPYLM models—one after 32 Gibbs sampling iterations (HPYLM-iter32), and the other after 100 iterations (HPYLM-iter100).

We again evaluated perplexity performance over these four data sets to investigate the scalability of perplexity experiments. The perplexity results in Table VIII indicate that the HPYLM scales well to larger training data. We obtained consistent reductions in perplexity over both the IKNLM and the MKNLM. This further strengthens the perplexity results of Section V-B. For two HPYLM models trained on ALL, we did not observe a significant difference in perplexity between HPYLM-iter32 and HPYLM-iter100.

Finally we trained three types of ARPA format trigram LMs—IKNLM, MKNLM, and HPYLM—on both ALL-WC

TABLE VIII
PERPLEXITY RESULTS ON *RT*06SEVAL USING DIFFERENT SCALE SIZES OF TRAINING DATA

Data	IKNLM	MKNLM	HPYLM	
MTRAIN	110.1	106.5	**101.2**	
MTRAIN+F	106.7	103.6	**97.6**	
ALL-WC	102.9	100.8	**95.1**	
ALL	107.0	105.2	**99.3**(iter32)	**98.9**(iter100)

TABLE IX
WER (%) RESULTS ON *RT*06SEVAL USING DIFFERENT SCALE SIZES OF TRAINING DATA

DATA	LMS	SUB	DEL	INS	WER
ALL-WC	IKNLM	14.6	10.0	2.6	27.3
	MKNLM	14.6	9.9	2.5	27.0
	HPYLM	**14.4**	**10.0**	**2.6**	**26.9**
ALL	IKNLM	14.5	9.7	2.7	27.0
	MKNLM	14.4	9.8	2.7	26.8
	HPYLM-iter32	**14.2**	**9.8**	**2.5**	**26.6**
	HPYLM-iter100	**14.2**	**9.8**	**2.6**	**26.5**

(a corpus of around 50 million word tokens) and ALL (a corpus of around 210 million word tokens) training data sets. Table IX shows the WER results of these three different LMs in the first decoding using `HDecode`. On ALL-WC, we see the HPYLM performs slightly better than the IKNLM and the MKNLM. Significance testing shows the reductions by the HPYLM are not significant. On ALL, however, we observed significant reductions in WER by using the HPYLM, with $p < 0.0002$ and $p < 0.004$ for reductions over the IKNLM and MKNLM, respectively. Once again, there is no significant difference in WER between HPYLM-iter32 and HPYLM-iter100.

E. Data Combination Versus Model Interpolation

Given several text corpora, i.e., those seven shown in Table XI, there are two different ways to estimate a language model. *Data combination* simply concatenates those seven corpora and trains a single language model on the combined corpus, without considering the differences between the corpora. This is the way in which we trained most LMs for the above experiments. *Model interpolation*, on the contrary, estimates seven separate language models on the corpora respectively, and linearly interpolates these seven LMs using some development data to optimize the interpolation weights. We have demonstrated the effectiveness of the HPYLM in the data combination style. Since model interpolation is commonly used in most state-of-the-art ASR systems, it is worthwhile for us to investigate the case of model interpolation.

We first investigated the experiments on *rt06seval*. We trained four sets of language models separately on MTRAIN, FISHER, WEBMEET, and WEBCONV in Table I, with each set using interpolated Kneser–Ney, modified Kneser–Ney, and hierarchical Pitman–Yor process smoothing respectively. For each type of smoothing method, we interpolated the four separate language models using a development set of the evaluation data from NIST RT05s *rt05seval* (with 2216 sentences, 16 282 word tokens). The final language models were obtained by

TABLE X
PERPLEXITY (PPL) AND WER (%) RESULTS OF COMPARING DATA COMBINATION AND MODEL INTERPOLATION ON *RT*06SEVAL TESTING DATA

Experiment	LMS	PPL	SUB	DEL	INS	WER
Model	IKNLM	83.3	13.5	9.8	2.4	25.7
Interpolation	MKNLM	83.3	13.5	9.8	2.4	25.8
for *rt06seval*	HPYLM	**81.1**	**13.7**	**9.8**	**2.4**	**25.9**

TABLE XI
STATISTICS OF CORPORA, AND THE PERPLEXITY RESULTS ON THE FOLD 1 OF THE SCENARIO AMI MEETINGS

Data Sets	#Word Tokens	Perplexity		
		IKNLM	MKNLM	HPYLM
AMI fold2-5	771,870	109.3	106.4	**102.3**
ICSI	650,171	258.7	243.0	**234.0**
NIST-a	119,989	299.7	283.9	**277.5**
ISL-MCL	119,093	327.5	308.5	**304.0**
h5etrain03v1	3,494,444	234.6	223.3	**210.6**
Fisher-03-p1+p2	21,235,716	221.2	210.9	**200.7**
Hub4-lm96	130,896,536	321.1	301.3	**289.1**

interpolating with the optimal weights. Table X shows the perplexity and WER results for model interpolation on *rt06seval*. In comparison to data combination results for ALL (perplexity in Table VIII and WER in Table IX), we find model interpolation has both lower perplexity and lower WER. Considering the results for model interpolation, the HPYLM has slightly lower perplexity, but statistically insignificant WER, than the IKNLM and the MKNLM.

We additionally carried out the experiments on the fivefold scenario AMI meetings in Table II to compare the two cases, using fold 2–5 as the training data and fold 1 as the testing data. We furthermore included six extra corpora in Table XI as the training data for LMs. For data combination, we reached a combined corpus of 157 million word tokens in total. The parallel training algorithm was used to infer an HPYLM on this combined corpus, with 100 Gibbs sampling iterations. For model interpolation, we trained IKNLMs, MKNLMs, and HPYLMs (with 100 iterations) separately, using the seven training corpora, respectively. Table XI shows the perplexity results, indicating that the HPYLM consistently has a lower perplexity than the IKNLM and the MKNLM on each LM component. We used a fourfold cross validation on folds 2–5 of the scenario AMI meetings to tune the optimal interpolated weights. Each time we took one fold from 2–5 as the development set, and the remaining three as the training data on which we trained the IKNLM, MKNLM, and HPYLM, respectively. The weights for the seven LM components were then optimized using the development set, for the IKNLM, MKNLM, and HPYLM, respectively. We considered the average of the accumulated weights as the final optimal weights, which were used to estimate an interpolated LM.

Table XII shows the perplexity and WER results on fold 1. It is not surprising to find that model interpolation is superior to data combination, because model interpolation weights the LM components of different domains to better match the testing data. Model interpolation provides significantly better results than data combination in perplexity and WER. We observed

TABLE XII
PERPLEXITY AND WER (%) RESULTS OF COMPARING DATA COMBINATION AND MODEL INTERPOLATION USING THE FOLD 1 OF THE SCENARIO AMI MEETINGS IN TABLE II AS THE TESTING DATA

Experiment	LMS	PPL	SUB	DEL	INS	WER
Data	IKNLM	168.6	22.2	10.7	5.7	38.6
Combination	MKNLM	163.9	22.0	10.8	5.6	38.5
for fold 1	HPYLM	**158.8**	**21.9**	**10.8**	**5.5**	**38.2**
Model	IKNLM	93.8	20.4	11.0	5.3	36.8
Interpolation	MKNLM	94.0	20.5	11.1	5.3	36.8
for fold 1	HPYLM	**91.4**	**20.4**	**11.2**	**5.1**	**36.7**

much higher perplexity results from data combination compared to model interpolation, due to the fact that a large portion of out-of-domain data (Hub4-lm96) was weighted identically to the in-domain meeting data in data combination. In either data combination or model interpolation, however, the HPYLM consistently has a lower perplexity result, and significantly $(p < 0.05)$ lower WERs than the IKNLM and the MKNLM (although the absolute reductions are small for model interpolation). This suggests that we can train separate HPYLMs on several different corpora, and then use the standard method to interpolate these separate HPYLMs. This further consolidates our claim that the HPYLM is a better smoothing method than the IKNLM and the MKNLM for practical ASR tasks. It is more desirable, however, for a method to automatically weight and interpolate several HPYLMs directly within the hierarchical Pitman–Yor process framework. Wood and Teh [35] have proposed a model within the hierarchical Pitman–Yor process framework for domain adaptation. This approach, however, can only deal with two components, the in-domain LM and the out-of-domain LM, respectively. Additionally the computational complexity should be considered when doing interpolation for large corpora.

VI. ANALYSIS AND DISCUSSIONS

A. Convergence

It is often expensive to train an HPYLM, especially when working with large training corpora as demonstrated in Table VII. Therefore, the convergence of HPYLM is an important factor. We trained an HPYLM using the data set MTRAIN in Table I. During each iteration, we collected the log likelihood over the training data, and the predictive log likelihood over the testing data *rt06seval*. Fig. 3 shows the convergence of likelihoods over training and testing data for the first 150 iterations. From this we can see that after about 20 iterations, the HPYLM has quickly converged to a lower predictive log likelihood value on the testing data, which roughly remains the same for further iterations. On the other hand, although it is slow to train an HPYLM on large corpora, we only need to train the model once and output an ARPA format LM, then apply it in an ASR system as a standard n-gram LM. We also observed that the likelihood over the training data decreases after more and more iterations, while the likelihood over the testing data increases, which means the generalization of the HPYLM improves.

The finding is further confirmed by from the experimental perplexity and WER results in Tables VIII and IX, respectively, for two HPYLM models, one from the 32nd iteration and the

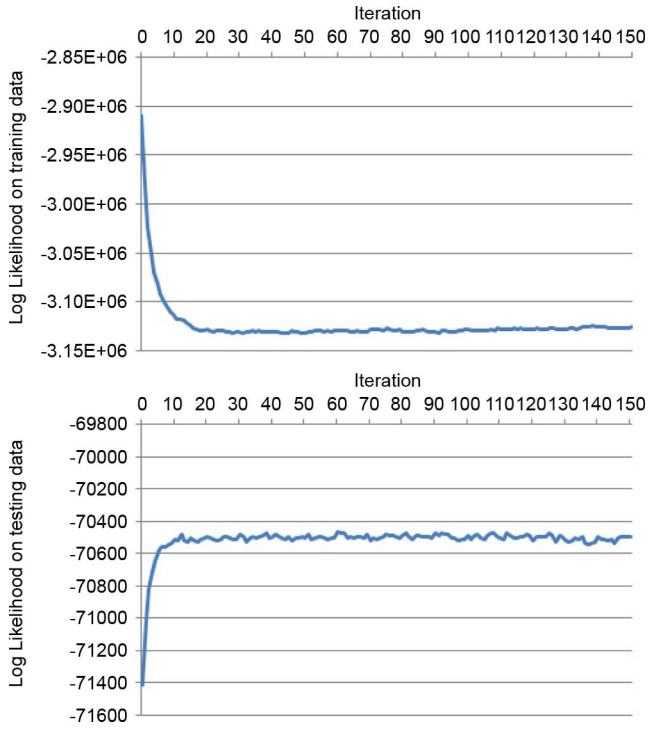


Fig. 3. Convergence of the HPYLM. The log likelihood over the training data MTRAIN (top), and the log likelihood over the testing data *rt06seval* (bottom), to investigate the convergence of the HPYLM with iterations.

other from the 100th iteration. There is no significant difference between these two models, which indicates normally we need only tens of iterations to infer an HPYLM model, since the posterior of the HPYLM is well-behaved.

B. Average Discounts

For a specific order of an n-gram LM, there is only one discount $d_{|\mathbf{u}|}$ for the IKNLM. The MKNLM, instead, has three different discount parameters, $d_{|\mathbf{u}|1}$, $d_{|\mathbf{u}|2}$, and $d_{|\mathbf{u}|3+}$, according to the counts of n-grams. This additional flexibility of discounting in the MKNLM has been proven to be superior in practice to the IKNLM [2]. We find from (12) that the HYPLM has a more relaxed discounting scheme, by allowing each context to use a different discounting value: $d_{|\mathbf{u}|}t_{\mathbf{u}w}$. It was noted by Teh [36] that the expected number of tables $t_{\mathbf{u}w}$ in a Pitman–Yor process scales as $O(c_{\mathbf{u}w}^{d_{|\mathbf{u}|}})$ where $c_{\mathbf{u}w}$ is the number of customers and $d_{|\mathbf{u}|}$ is the discount parameter of the Pitman–Yor process. Therefore, the actual amount of discount, $d_{|\mathbf{u}|}t_{\mathbf{u}w}$, will grow slowly as the count $c_{\mathbf{u}w}$ grows.

To demonstrate the rich-get-richer property of discounting in the HPYLM, we investigate an HPYLM trained for 300 iterations using MTRAIN (meeting data) in Table VII, and plot average discounts as a function of trigram counts in Fig. 4. We additionally plot on the bottom of Fig. 4 the *counts of counts* statistics used to do the averaging over discounts. The counts of counts histogram exhibits a long-tailed distribution: most words occur with lower counts, while only a few words occur frequently. We find that the growth of average discounts in the HPYLM is sublinear, with respect to counts of trigrams. For a comparison, the IKNLM uses a single disccounting parameter: $d_3 = 0.8083$, and the MKNLM uses three different discounts: $d_{31} = 0.8083$, $d_{32} = 1.1428$, $d_{33+} = 1.3551$. It is interesting to note that discounts in the HPYLM for trigrams with large values of counts are also surprisingly large, which reminds us to question whether or not it is enough to use only one discount parameter $d_{|\mathbf{u}|3+}$ in the MKNLM for all n-grams with three or more counts. Finally we note that similar counts may have substantial differences in average discounts, especially in the case of larger counts. This may arise from local effects during the sample-based inference of the seating arrangements—more likely in a restaurant with more customers (larger counts)—-and also because discounts for larger counts are averaged over fewer trigrams (count of counts changes inversely with counts), which makes it look less smoothed for large counts.

C. Validation of Parallelization Approximation

To measure experimentally the errors introduced by the parallel training algorithm, we trained two HPYLMs of order 3 on FISHER shown in Table I for 100 iterations, one without parallelism, while the other with the parallel training algorithm introduced in Section IV-C. We evaluated the two HPYLMs for the transcription of *rt06seval*, recording the perplexity results for each iteration. Fig. 5 shows that there is no statistically significant difference between perplexity results of these two HPYLMs, which indicates that the error caused by the approximation in parallel training algorithm can be ignored. We also find no significant difference in the WER [37].

D. Learning Curve

To further verify that the HPYLM is a suitable choice for language modeling when there is a large amount of data, we investigated the learning curve of the HPYLM with respect to the amount of training data. We considered ALL, a corpus of 211 million word tokens consisting of data sets No.1–8 in Table I. We first randomly reordered the corpus, and then varied the training set size between 1 million and 211 million words by gradual increments to obtained training sets of 1, 10, 25, 50, 100, 200, and 211 million words. We trained the IKNLM, MKNLM, and HPYLM using the CV condition on these training sets, respectively, and evaluated the language models on the RT06s test data (*rt06seval*), as in Section V-B. Fig. 6 shows the learning curve of perplexity results as the amount of training data increases. We see that, although there is a runtime overhead, the HPYLM consistently outperforms the IKNLM and the MKNLM as the size of training data increases. Moreover, the reductions by the HPYLM become larger with larger amount of training data available.

E. Comparison to Other Models

In our experiments, the HPYLM produces consistently better perplexity results than the MKNLM. This observation is in contrast to the finding in [12], where the HPYLM performs slightly worse than the MKNLM in terms of perplexity. We argue that there are two potential reasons for this. First, [12] used conjugate gradient descent in the cross-entropy on the validation set to determine the discounting parameters for the IKNLM and the MKNLM, which is a different approach to that used in most standard language model toolkits such as SRILM [38]. We used

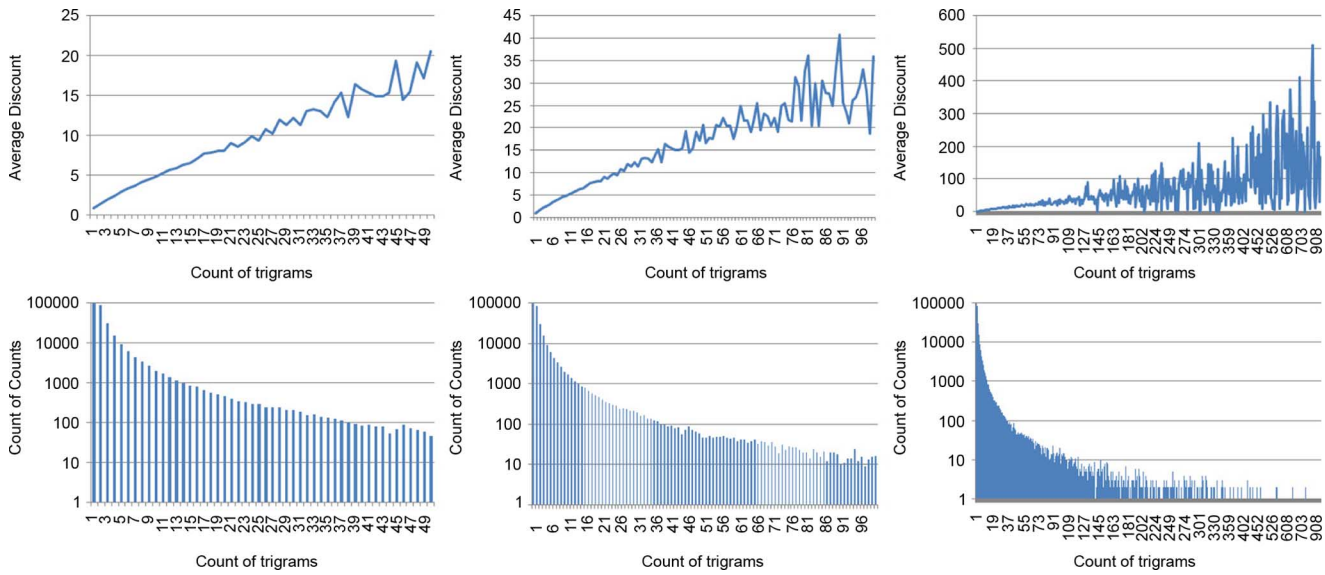


Fig. 4. Average discounts as a function of trigram counts (top), and counts of trigram counts (bottom) in the HPYLM trained on the data set MTRAIN, with different scales for horizontal axis: first 50 counts (left), first 100 counts (middle), and first 1000 counts(right).

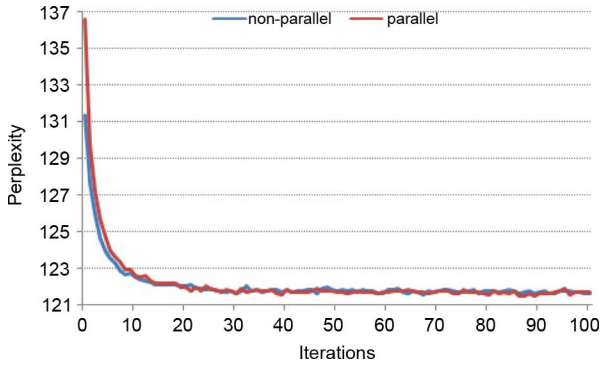


Fig. 5. Perplexity results by iterations, trained on FISHER and tested on RT06s evaluation data *rt06seval* for the HPYLM, to validate parallelization approximation.

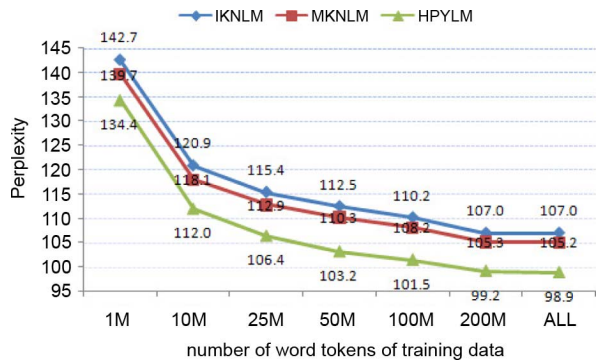


Fig. 6. Learning curve of perplexity results on RT06s evaluation data *rt06seval* for the HPYLM with respect to the amount of training data.

in this paper the SRILM toolkit to build reasonable baseline results and evaluate our various LMs. Second, the difference in the data and the implementation may be another reason. Based on the discussion in Section VI-B, however, we believe that it makes sense for the HPYLM to outperform the MKNLM, because of its more flexible discounting scheme for each different context.

There are different interpretations for the unusual modified counts for lower order n-grams in the IKNLM. Kneser and Ney [7], and Chen and Goodman [2], derived the modified counts for lower order n-grams in terms of preserving marginal word distribution constraints. Goodman [34] justified this from a Bayesian view of a maximum entropy model with an exponential prior, and clearly explained why the Kneser–Ney smoothing has that particular form including the modified counts for lower order n-grams. The maximum entropy model, to which the back-off version of Kneser–Ney smoothing forms an approximation, also preserves the marginal constraints. The HPYLM is a full Bayesian interpretation of interpolated Kneser–Ney as approximate inference in a hierarchical Bayesian model consisting of Pitman–Yor processes [12]. The modified counts for lower order n-grams, $c_{\mathbf{u}w\bullet}$, are derived directly from Chinese restaurant processes and different from those in the IKNLM, subject to the following relationships among $c_{\mathbf{u}w\bullet}$ and $t_{\mathbf{u}w\bullet}$ [12]:

$$\begin{cases} t_{\mathbf{u}w\bullet} = 0 & \text{if } c_{\mathbf{u}w\bullet} = 0; \\ 1 \le t_{\mathbf{u}w\bullet} \le c_{\mathbf{u}w\bullet} & \text{if } c_{\mathbf{u}w\bullet} > 0; \end{cases} \qquad c_{\mathbf{u}w\bullet} = \sum_{\mathbf{u}':\pi(\mathbf{u}')=\mathbf{u}} t_{\mathbf{u}'w\bullet}$$

where $t_{\mathbf{u}'w\bullet}$ is the number of tables seated by $c_{\mathbf{u}'w\bullet}$ customers in the child restaurant of $\mathbf{u}$. The value of $t_{\mathbf{u}'w\bullet}$, and consequently the modified count $c_{\mathbf{u}w\bullet}$, are naturally determined by the seating arrangements induced from Chinese restaurant processes. If we constrain $t_{\mathbf{u}'w\bullet} = 1 \text{if } c_{\mathbf{u}w\bullet} > 0$, the modified count $c_{\mathbf{u}w\bullet}$ is the same as that in the IKNLM. However, the HPYLM also satisfies marginal constraints when the strength parameter $\theta_m = 0$ for all $m < n$, as proved in [36].

The structural Bayesian language model by Yaman *et al.* [33] shares similar ideas to the hierarchical Dirichlet language model [10] (although it is estimated by a MAP approach), in that both models assemble n-grams in a tree structure and

use the Dirichlet distribution as the prior to smooth the empirical counts. In comparison to the HPYLM, however, both models suffer from performance improvements compared to state-of-the-art smoothing methods such as IKN and MKN, for lack of one important issue for language model smoothing—absolute discounting.

VII. Conclusion

In this paper, we present the application of hierarchical Pitman–Yor process language models on a large-vocabulary ASR system for conversational speech, using reasonably large corpora. We show comprehensive experimental results on multiparty conversational meeting corpora, with the observation that the HPYLM outperforms both the IKNLM and the MKNLM.

Overall, we hope to convey our judgment that it is feasible, and worthwhile, to use the HPYLM for applications in large-vocabulary ASR systems. In detail, the conclusions we make in this paper are as follows: 1) the HPYLM provides an alternative interpretation, Bayesian inference, to language modeling, which can be reduced to interpolated Kneser–Ney smoothing [12]; 2) the HPYLM provides a better smoothing algorithm for language modeling in practice, which has better perplexity and WER results than both the IKNLM and the MKNLM, consistent and significant; 3) HPYLM training converges in relatively quickly; 4) a parallel training scheme makes it possible to estimate models in the case of large training corpora and large vocabularies.

The main contributions of the HPYLM work in this paper include the introduction of a novel Bayesian language modeling technique to the ASR community, and the experimental verification on the task of large-vocabulary ASR for conversational speech in meetings. We have demonstrated that it is feasible to infer a hierarchical non-parametric Bayesian language model from a large corpus, thus making it practical to use for large vocabulary speech recognition or machine translation. Our experimental results have shown that any approximations resulting from the parallel algorithm have a negligible effect on performance. Overall, the HPYLM results in significantly improved accuracy compared with the current state-of-the-art (IKNLM/MKNLM). The resulting language model may be interpreted as a smoothed n-gram model, can be implemented in a standard way (e.g., using an ARPA format language model file), and may be used in place of other smoothed n-gram language models.

Appendix A
Program and Scripts for the HPYLM

The executable program (hpylm), the corresponding Python scripts, and part of the text data used in this paper to train and test an HYPLM, are available from http://homepages.inf.ed.ac.uk/s0562315/.

TABLE XIII
Meeting IDs for the 5 Folds of the AMI Scenario Meetings. The Initial Letter "S" Represents the Scenario Meetings

FOLD	No.	MEETINGS
1	32	S2004[A-D] S2005[A-D] S2007[A-D] S2013[A-D] S2014[A-D] S3007[A-D] S3008[A-D] S3009[A-D]
2	28	S2006[A-D] S2009[A-D] S2011[A-D] S2016[A-D] S3005[A-D] S3010[A-D] S3011[A-D]
3	24	S1009[A-D] S2003[A-D] S2012[A-D] S2015[A-D] S3006[A-D] S3012[A-D]
4	28	S1007[A-D] S1008[A-D] S2002[A-D] S2008[A-D] S2010[A-D] S3003[A-D] S3004[A-D]
5	25	S1000[A-D] S1001[A-D] S1002[B-D] S1003A S1003C S1003D S1004[A-D] S1005[A-C] S1006[A-D]

Appendix B
The Fivefold Cross-Validation Split-Up

We include below in the Table XIII the meeting identities from the scenario AMI Meeting Corpus for the fivefold cross validation in Section V.

Appendix C
Matched-Pair Significance Test

We use in this paper the matched-pair significance test method [42] for significance testing of WER results in the ASR experiments.

Suppose **baseline** and **newsystem** are the filtered hypothese files, generated by the NIST scoring tool **sclite**. We use the script **compare − sclit** from the SRILM toolkit [38] to compute the matched pairs:

compare − sclite − rrefs − h1baseline − h2newsystem

Let M be the number of times the two systems' hypotheses differ, and N be the number of times new system improves upon baseline. We compute the probability that a fair coin would have come out heads at least N times in totally M trials: $P(k \geq N|M)$ as the significance p-value, with $p < 0.01$ to be significant, and $p < 0.05$ to be significant but weak.

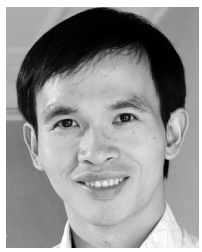
Acknowledgment

The authors would like to thank the anonymous reviewers for their excellent comments, the AMI-ASR team for providing the baseline ASR system for experiments, and Y. W. Teh for helpful discussions and sharing his Matlab codes to double-check our implementation.

References

[1] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge, U.K.: MIT Press, 1997.

[2] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Comput. Speech Lang.*, vol. 13, no. 4, pp. 359–393, 1999.
[3] J. T. Goodman, "A bit of progress in language modeling," *Comput. Speech Lang.*, pp. 403–434, 2001.
[4] I. J. Good, "The population frequencies of species and the estimation of population parameters," *Biometrika*, vol. 40, no. 3/4, pp. 237–264, 1953.
[5] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recogniser," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-35, no. 3, pp. 400–401, Mar. 1987.
[6] F. Jelinek and R. L. Mercer, E. S. Gelsema and L. N. Kanal, Eds., "Interpolated estimation of Markov source parameters from sparse data," in *Proc. Workshop Pattern Recognition in Practice*, 1980, pp. 381–397.
[7] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," *Proc. ICASSP'95*, pp. 181–184, 1995.
[8] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.
[9] J. Blitzer, A. Globerson, and F. Pereira, "Distributed latent variable models of lexical co-occurrences," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, 2005.
[10] D. J. C. MacKay and L. C. B. Peto, "A hierarchical Dirichlet language model," *Natural Lang. Eng.*, vol. 1, no. 3, pp. 1–19, 1994.
[11] S. J. Goldwater, T. L. Griffiths, and M. Johnson, "Interpolating between types and tokens by estimating power-law generators," *Adv. Neural Inf. Process. Syst. 18*, 2006.
[12] Y. W. Teh, "A hierarchical Bayesian language model based on Pitman–Yor processes," in *Proc. Annu. Meeting ACL*, 2006, vol. 44.
[13] J. A. Bilmes and K. Kirchhoff, "Factored language models and generalized parallel backoff," in *Proc. HLT/NACCL*, 2003, pp. 4–6.
[14] P. Xu, A. Emami, and F. Jelinek, "Training connectionist models for the structured language model," in *Proc. Empirical Methods Natural Lang. Process. (EMNLP'03)*, 2003.
[15] S. Huang and S. Renals, "Unsupervised language model adaptation based on topic and role information in multiparty meetings," in *Proc. Interspeech'08*, Sep. 2008, pp. 833–836.
[16] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*, 2nd ed. London, U.K.: Chapman & Hall/CRC, 2004.
[17] S. Renals, T. Hain, and H. Bourlard, "Recognition and interpretation of meetings: The AMI and AMIDA projects," in *Proc. IEEE Workshop Autom. Speech Recognition Understanding (ASRU'07)*, 2007, pp. 238–247.
[18] S. Huang and S. Renals, "Hierarchical Pitman–Yor language models for ASR in meetings," in *Proc. IEEE ASRU'07*, Dec. 2007, pp. 124–129.
[19] C. E. Antoniak, "Mixtures of Dirichlet processes with application to non-parametric problems," *Ann. Statist.*, vol. 2, no. 6, pp. 1152–1174, 1974.
[20] T. S. Ferguson, "A Bayesian analysis of some nonparametric problems," *Ann. Statist.*, vol. 1, no. 2, pp. 209–230, 1973.
[21] J. Sethuraman, "A constructive definition of Dirichlet priors," *Statist. Sinica*, vol. 4, pp. 639–650, 1994.
[22] Y. W. Teh, "Dirichlet processes," in *Encyclopedia Mach. Learn.* . : , 2007.
[23] J. Pitman and M. Yor, "The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator," *Ann. Probability*, vol. 25, no. 2, pp. 855–900, 1997.
[24] J. Pitman, "Exchangeable and partially exchangeable random partitions," *Probability Theory and Related Fields*, vol. 102, pp. 145–158, 1995.
[25] D. Aldous, "Exchangeability and related topics," in *École d'Été de Probabilités de Saint-Flour, ser. XIII–1983*. Berlin, Germany: Springer, 1985, pp. 1–198.
[26] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," *Mach. Learn.*, vol. 50, no. 1, pp. 5–43, Jan. 2003.
[27] B. J. Frey and N. Jojic, "A comparison of algorithms for inference and learning in probabilistic graphical models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 9, pp. 1392–1416, Sep. 2005.
[28] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Mach. Learn.*, vol. 37, no. 2, pp. 183–233, 1999.
[29] E. B. Sudderth, "Graphical models for visual object recognition and tracking," Ph.D. dissertation, Mass. Inst. of Technol., Cambridge, MA, 2006.
[30] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependences in stochastic language modelling," *Comput. Speech Lang.*, vol. 8, no. 1, pp. 1–38, 1994.
[31] 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," *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, 1992.
[32] A. Nadas, "Estimation of probabilities in the language model of the IBM speech recognition system," *IEEE Trans. Acoust.. Speech, Lang. Process.*, vol. ASSP-32, no. 4, pp. 859–861, Aug. 1984.
[33] S. Yaman, J.-T. Chien, and C.-H. Lee, "Structural Bayesian language modeling and adaptation," in *Proc. Interspeech'07*, Antwerp, Belgium, Aug. 2007, pp. 2365–2368.
[34] J. Goodman, "Exponential priors for maximum entropy models," in *Proc. HLT-NAACL*, 2004, pp. 305–311.
[35] F. Wood and Y. W. Teh, "A hierarchical nonparametric Bayesian approach to statistical language model domain adaptation," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2009, vol. 12, pp. 607–614.
[36] Y. W. Teh, "A Bayesian interpretation of interpolated Kneser–Ney," School of Computing, National Univ. of Singapore, 2006, Tech. Rep. TRA2/06.
[37] S. Huang and S. Renals, "A parallel training algorithm for hierarchical Pitman–Yor process language models," in *Proc. Interspeech'09*, Sep. 2009, pp. 2695–2698.
[38] A. Stolcke, "SRILM—An extensible language modeling toolkit," in *Proc. ICSLP'02*, Denver, CO, Sep. 2002, pp. 901–904.
[39] V. Wan and T. Hain, "Strategies for language model web-data collection," in *Proc. ICASSP'06*, Toulouse, France, May 2006, pp. 1069–1072.
[40] J. Carletta, "Unleashing the killer corpus: Experiences in creating the multi-everything AMI meeting corpus," *Lang. Res. Eval. J.*, vol. 41, no. 2, pp. 181–190, 2007.
[41] T. Hain, L. Burget, J. Dines, G. Garau, M. Karafiat, M. Lincoln, J. Vepa, and V. Wan, "The AMI system for the transcription of speech in meetings," in *Proc. ICASSP'07*, Apr. 2007, pp. 357–360.
[42] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 2009.



Songfang Huang (S'08) received the B.Sc. degree from North China Electric Power University, Beijing, China, in 2002, and the M.Sc. degree from Peking University, Beijing, China, in 2005. He is currently pursuing the Ph.D. degree at the Centre for Speech Technology Research, School of Informatics, University of Edinburgh, Edinburgh, U.K.

He was a Co-op Researcher at the IBM T. J. Watson Research Center, Yorktown Heights, NY, from July to November 2008, and a Research Associate for the AMIDA project at the Centre for Speech Technology Research, School of Informatics, University of Edinburgh, from December 2008 to September 2009. Since November 2009, He has been a Postdoctoral Researcher at the IBM T. J. Watson Research Center. His research interests mainly concern the application of machine learning algorithms to automatic speech recognition, natural language processing, and statistical machine translation.

Steve Renals (M'91) received the B.Sc. degree from the University of Sheffield, Sheffield, U.K., in 1986, and the M.Sc. and Ph.D. degrees from the University of Edinburgh, Edinburgh, U.K., in 1987 and 1991, respectively.

He is a Professor of speech technology in the School of Informatics and Director of the Centre for Speech Technology Research, University of Edinburgh. He held postdoctoral fellowships at the International Computer Science Institute, Berkeley, CA, (1991–1992) and at the University of Cambridge, Cambridge, U.K. (1992–1994). He was a member of academic staff at the University of Sheffield for nine years as a Lecturer (1994–1999), then Reader (1999–2003). He is an Associate Editor of the ACM Transactions on Speech and Language Processing. His main research areas are in spoken language processing and multimodal interaction, and he has over 150 publications in these areas.