Available online at www.sciencedirect.com

SciVerse ScienceDirect

Speech Communication xxx (2012) xxx–xxx

SPEECH COMMUNICATION

www.elsevier.com/locate/specom

Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model ☆

Anoop Deoras [a,*], Tomáš Mikolov [b], Stefan Kombrink [b], Kenneth Church [c]

[a] Microsoft Corporation, 1065 La Avenida. Mountain View, CA 94043, United States
[b] Speech@FIT, Brno University of Technology, Brno, Czech Republic
[c] IBM T.J. Watson Research Center, Yorktown Heights, NY, United States

Received 30 December 2011; received in revised form 23 July 2012; accepted 3 August 2012

Abstract

In this paper, we present strategies to incorporate long context information directly during the first pass decoding and also for the second pass lattice re-scoring in speech recognition systems. Long-span language models that capture complex syntactic and/or semantic information are seldom used in the first pass of large vocabulary continuous speech recognition systems due to the prohibitive increase in the size of the sentence-hypotheses search space. Typically, n-gram language models are used in the first pass to produce N-best lists, which are then re-scored using long-span models. Such a pipeline produces *biased* first pass output, resulting in sub-optimal performance during re-scoring. In this paper we show that computationally tractable *variational approximations* of the long-span and complex language models are a better choice than the standard n-gram model for the first pass decoding and also for lattice re-scoring.

Keywords: Long-span language models; Recurrent neural networks; Speech recognition; Decoding

1. Introduction

In state-of-the-art recognition systems today, language model (LM) is still restricted to a simple n-gram like model, where the distribution of the predicted word depends on the previous $(n-1)$ words. i.e. if w_i is the hypothesized word, then its probability given the past is approximated by $P(w_i|w_{i-1},\ldots,w_{i-n+1})$. A major reason for using such simple LMs, besides the ease of estimating them from text, is computational efficiency. The search space (time) in large vocabulary continuous speech recognition (LVCSR) decoding is governed by the number of distinct "equivalent" histories, i.e. the number of unique probability distributions needed to account for all possible histories, and it grows nearly exponentially with n-gram order. So it is customary to limit n to 3 or 4.

Long-span LMs, capturing syntactic, semantic, discourse or other long-distance dependencies, are much more powerful than low order n-gram LMs and will hopefully outperform the latter models in terms of metrics such as perplexity and word error rate. However, incorporating long-span LMs for decoding is a computationally challenging problem. Hence the standard practice is to carry out a first pass of decoding using, say, a 3-gram LM to generate a *lattice*, and to rescore only the hypotheses in the lattice with a higher order LM, such as a 4- or 5-gram. But even the search space defined by a lattice is intractable for many long-span LMs. In such cases, only the N-best full-utterance hypotheses from the lattice are extracted for evaluation by the long-span LM. Typically, N is a few thousand, if not a few hundred.

☆ Some initial results based on the proposed idea have previously been published in (Deoras et al., 2011b). This work was carried out while the first and last authors were at Johns Hopkins University, Baltimore, MD.

* Corresponding author. Tel.: +1 650 693 3799; fax.: +1 425 823 8659.
E-mail addresses: Anoop.Deoras@microsoft.com, adeoras@jhu.edu (A. Deoras), imikolov@fit.vutbr.cz (T. Mikolov), kombrink@fit.vutbr.cz (S. Kombrink), kwchurch@us.ibm.com (K. Church).

0167-6393/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.specom.2012.08.004

Using an n-gram LM in the first pass however produces hypotheses more representative of such a model. If the gap between the starting n-gram LM and the long-span LM is large then typically re-scoring of these first pass N-best hypotheses can lead to sub-optimal performance. The n-gram LM may assign such a low score to good hypotheses that they fail to appear among the N-best. If such hypotheses would have eventually surfaced to the top due to the long-span LM, their loss is attributable to the "bias" of the N-best list towards the first pass LM. For this reason, we seek ways to incorporate information from long-span LMs directly into first pass decoding as well as during lattice re-scoring.

We propose to approximate the long-span model using *variational approximation* techniques. Given a long-span model P, possibly a sophisticated LM with complex statistical dependencies, we will seek a simple and computationally tractable model Q^* that will be a good surrogate for P. Specifically, among all models Q of a chosen family $\mathcal{Q}$ of tractable models, we will seek the one that minimizes the Kullback–Leibler divergence (Cover and Thomas, 1991) from P. We will use this model for first pass decoding, which in turn produces richer lattices and faithful N-best lists. We then either rescore these lattices with bigger models or deploy the full blown model on the N-best lists extracted from the first pass recognition. Thus the N-best lists that are extracted from the first pass decoding are no more biased towards the weaker baseline model.[1] They are in-fact biased towards the model with which we wish to do the re-scoring. This technique thus helps to produce a refined and less biased search space for second pass decoding such that re-scoring on it becomes much more effective.

Examples of P include computationally powerful LMs outside the family of finite state machines, such as recurrent neural network language models (Mikolov et al., 2010), random forests (Xu, 2005) and structured language models (Chelba and Jelinek, 2000; Roark, 2001). We will approximate P with a Q^* from the family $\mathcal{Q}$ of finite state machines, mainly comprising of n-grams.[2] The choice of $\mathcal{Q}$ is driven by decoding capabilities.

Previously, researchers have addressed the problem of approximating stochastic context free grammar (SCFG) by probabilistic finite state automaton (PFA) for the use in speech recognition systems. For instance, Stolcke and Segal (1994) demonstrated how one could precisely compute bi-gram probabilities from a SCFG for use in the speech recognition systems. It has, however, been difficult to find out how well the resulting models approximate the SCFGs and more importantly, what should be the correct measure? Nederhof and Satta, 2004 argued that the right way to obtain an approximate PFA from SCFG would be to explicitly minimize the Kullback–Leibler divergence (KLD) between the two probability distributions (i.e. SCFG and PFA), as KLD is a natural distance metric between any two probability distribution functions. In their later work (Nederhof, 2005) they showed how one could also search for a PFA such that the KLD with SCFG would be minimal over all possible PFAs. They, however, did not report any results.

The theoretical foundation of our work is similar in spirit to the ones mentioned above. In our work, we, however, show for the first time that using Monte Carlo techniques, it is possible to approximate near Turing machine like models (Elman, 1990) with higher order PFAs and use them directly in the first pass speech recognition systems and/or re-scoring the word lattices. We show how the approximated PFAs are better than the PFAs estimated from the training data, thus improving the overall speech recognition performance. We demonstrate the working of our proposed idea on many state-of-the-art large vocabulary continuous speech recognition setups and find out some very interesting statistics about the novel n-grams learnt during the approximation process.

In our other work (Deoras and Jelinek, 2009; Deoras et al., 2011a; Deoras, 2011, chap. 4), we have proposed iterative decoding, a hill climbing technique to tackle the problem of incorporating long span language models for recognition. In it we have proposed to replace the global search problem (search for the most likely hypothesis under long-context LMs) by series of many convenient local search problems. Local search problems were defined so that long span models could be deployed without needing to approximate the models. Such a method thus approximates the search space (word lattices) but keeps the re-scoring model (long-span models) intact. In this paper, we propose to approximate the model instead and keep the search space intact. Thus these two methods offer two complementary approaches to solve an important problem of incorporating long-context LM for decoding.

The rest of the paper is organized as follows. Section 2 provides details about the proposed methodology for language modeling using ideas from variational approximation. Section 3 discusses some of the complex long span LMs and Section 3.1 briefly describes our candidate LM – recurrent neural network language model. Section 4 describes our experimental setup and presents a number of results. Finally, a summary of the paper with some remarks is presented in Section 5. Sample simulations of text corpus obtained using our proposed technique is shown in Appendix A.

2. Variational approximation of a model

There are many popular methods of approximate inference, among which variational inference has gained popularity due to its simplicity (Jordon, 1998, pp. 105–162). Such methods are necessary when exact inference is intractable. In variational inference, a surrogate model

[1] Here we are referring to a weak n-gram LM which is used in the decoder to produce word lattices.

[2] See (Allauzen et al., 2003) for a representation of n-gram LM as a finite state machine (FSM).

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

characterized by the distribution $Q \in \mathcal{Q}$ is chosen to replace a complex model, characterized by the distribution P, such that inference under Q becomes much more tractable. Q should be found out such that it is closest to P in some sense. Since these models are probability distributions, a very natural choice of distance metric is Kullback–Leibler divergence (KLD). The surrogate[3] model Q is thus chosen such that among all the distributions in the family of the chosen parameterization, $\mathcal{Q}$, it has the minimum KLD with the complex distribution P. Thus if we decide on a family of distributions, $\mathcal{Q}$, then the surrogate distribution is found out by solving the following optimization problem (Eq. 1):

$$Q^* = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} D(\mathbf{P}\|\mathbf{Q}) = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \sum_{\cdot} P(\cdot) \ln \frac{P(\cdot)}{Q(\cdot)}$$
$$= \underset{Q \in \mathcal{Q}}{\operatorname{argmax}} \sum_{\cdot} P(\cdot) \ln Q(\cdot) \quad (1)$$

where $D(\mathbf{p}\|\mathbf{q}) = \sum_{x \in \mathcal{X}} p(x) \ln \frac{p(x)}{q(x)}$ is the KL Divergence between the probability mass functions p and q.

In our work, we choose $\mathcal{Q}$ to be the family of distributions parameterized by n-grams[4] i.e. we want to learn a model, Q, parameterized by n-grams, that is closest to P in the sense of Kullback–Leibler divergence. Under some mild conditions, Q^* is the n-dimensional marginal of P.

A natural question is whether Q^* is simply the n-gram model M estimated from the same (LM training) text that P was estimated from. Not surprisingly, the answer is negative. For one, even if both P and M were estimated from the same text, P may have been estimated with a different criterion than maximum likelihood (ML), so that its n-gram marginals may not agree with M, even after differences due to smoothing are ignored. Even if P is the ML estimate from a rich family $\mathcal{P}$ that contain $\mathcal{Q}$ as a subset, additional assumptions must hold about $\mathcal{P}$ for Q^* to be the same as M.

Fig. 1 illustrates the idea pictorially. $\mathcal{P}$ is the family of distributions which make the problem of decoding in ASR, computationally challenging, while $\mathcal{Q}$ is the family of distributions which make the same problem tractable. Choosing any $Q \in \mathcal{Q}$ can although make the decoding problem a tractable one, we are, however, interested in the model Q^*, which not only lies in $\mathcal{Q}$, but is also closest to $P(\in \mathcal{P})$. In our work, $\mathcal{Q}$ is the family of all n-gram language models. Model M is the maximum likelihood n-gram model, estimated from the same training data from which P is built.

But if P is indeed a long-span LM, then computing its n-dimensional marginal could also be computationally prohibitive. Often, and for our choice of P in Section 3.1, it is impossible to do so. This is mainly due to the intractable summation over all the words of the vocabulary for all those positions in the history, which fall outside the chosen

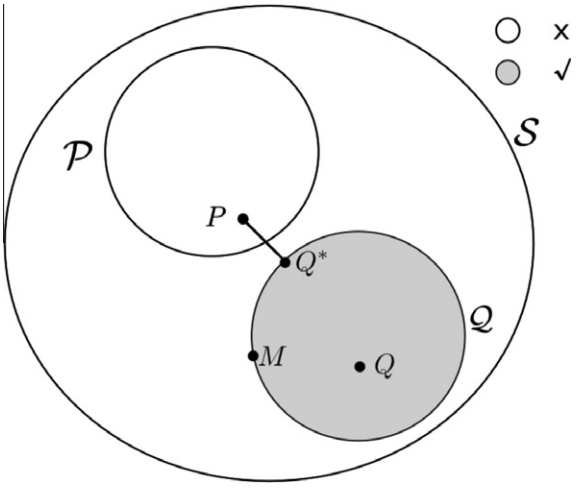

Fig. 1. Pictorial representation of family of distributions. $\mathcal{P}$ is the family of distributions such that under any distribution (language model) possible in this family, the ASR decoding problem becomes computationally challenging (a × symbol in the figure connotes this notion), while $\mathcal{Q}$ is the family of distributions such that under any distribution possible in this family, the ASR decoding problem becomes tractable (a √ symbol in the figure connotes this notion). In our work, $\mathcal{Q}$ is the family of all n-gram language models. Model M is the maximum likelihood n-gram model, estimated from the same training data from which P is built. The task then is to find Q^* such that it is closest to P.

n-gram window. So how does one proceed? For any P that is a generative model of text, the minimizer of (1) may be approximated via sampling (Geman and Geman, 1984; Bishop, 2006, p. 542). Thus by treating P as the generative model, we generate huge samples of data and then estimate a maximum likelihood based n-gram model. As we will show, such a model will minimize the KL Divergence with the long-span model and hence will be the faithful model in the chosen family of distributions. Details about this procedure follows in Section 2.1.

In our work, we call the technique of building an n-gram model from the data generated under some long span model as variational approximation mainly because the resulting approximate model, under certain conditions, minimizes the KL Divergence with the long-span model. This idea of finding a surrogate tractable model by minimizing the KL Divergence with the intractable parent model is similar in spirit to the variational inference idea used in machine learning, hence the choice of the terms *variational approximation* in our paper.

At this point, we can motivate the basic idea behind sampling for obtaining a tractable model from the long span language model. Let us say that in our training data we observe following two sentences:

- `I go to work on tuesday`
- `I go to work on wednesday`

and let us say that we also see the following sentence

- `today is tuesday`

[3] We use P and Q interchangeably for the model or the distribution.

[4] All the distributions in this family have the same order i.e. choice of n and this is dependent on the decoding capabilities.

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

but we do not see the sentence:

- `today is wednesday`

If such is our setup then the probability of the word `wednesday` given the context `today is` will not be robustly estimated (by the conventional *n*-gram LM) due to lack of observation in the training data. However, if some complex language model is able to *cluster* the words `tuesday` and `wednesday` based on the fact that they follow the same context i.e. `I go to work on`, then it is likely that such a model would predict both `tuesday` and `wednesday` given the context `today is`. If such a complex model is now used as a generative model, then it would possibly generate not only `tuesday` but also `wednesday` given the context of words `today is`. Thus such a technique would increase the coverage of *n*-grams and essentially improve the performance of language models in speech recognition systems.

Samples of some simulations that were produced for our experiments can be seen in Appendix A. In this Appendix we have contrasted the samples produced by a long span language model viz. recurrent neural network LM, with that by a short span language model, viz. an *n*-gram language model. It is interesting to see that for the same initial prefix, the samples produced by long span language model are more plausible than that by its short span counterpart.

2.1. Monte Carlo sampling

We *simulate* text using the distribution P, the one which underlies any generative long span language model. Given the start-of-sentence symbol <s>, we sample the next word from the probability distribution conditioned on <s>, and continue generating words conditioned on already generated words, i.e. given the sequence $w_1 w_2 \ldots w_{l-1}$ of words so far, the *lth* word is sampled from $P(\cdot|w_1,\ldots,w_{l-1})$, conditioned on the entire past.

At any stage, for some context h, generated so far, we query our long span LM to output a probability distribution, $P(w|h)$, over all the words w in the vocabulary $\mathcal{V}: p_1, p_2, \ldots, p_{|\mathcal{V}|}$. We then form cumulative distribution from the probability masses over all the words of the vocabulary. Cumulative distribution groups the words in bins taking values between 0 and 1. Thus the first word of the vocabulary falls between 0 and p_1, second word falls between p_1 and $p_1 + p_2$, so on and so forth. The last word falls between $p_{|V|-1}$ and 1. Once the words are binned, we randomly generate a real number between 0 and 1 using a uniform random generator. Depending upon the value generated, we find out the bin it belongs to and output the corresponding word. It should be clear that the probability of selecting any word in this sampling procedure, is equal to the probability of its occurrence given the context (according to the LM). The sampled word then forms part of the predicting history and a new word is sampled. The procedure is carried out until the sampled corpus size reaches a predetermined value.[5] In the special case of RNN-LMs which use class hierarchy in the output layer (Mikolov et al., 2011c), we first sample the classes given the history using the above sampling procedure and then given the sampled class, we sample the words. Since the number of classes are much smaller than the total vocabulary, sampling words from a class based language model is much faster.

Our estimate of Q^* is simply an *n*-gram model $\widehat{Q}^*$ based on this synthetically generated text.

If P is stationary and ergodic,[6] then the simulated corpus, L, will have the underlying distribution P, and then from the consistency of the maximum likelihood estimator (Bickel and Doksum, 1977), we can show that by solving (1), we essentially find out the maximum likelihood estimate, $\widehat{Q}^*$, in the *n*-gram family, based on the simulated corpus as shown below:

$$\begin{aligned} Q^* &= \operatorname*{argmin}_{Q\in\mathcal{Q}} D(\mathbf{P}\|\mathbf{Q}) = \operatorname*{argmin}_{Q\in\mathcal{Q}} \sum_{X\in\mathcal{X}} P(X)\ln\frac{P(X)}{Q(X)} \\ &= \operatorname*{argmax}_{Q\in\mathcal{Q}} \sum_{X\in\mathcal{X}} P(X)\ln Q(X) \\ &= \operatorname*{argmax}_{Q\in\mathcal{Q}} \lim_{L\to\infty} \sum_{l=1}^{L} \frac{1}{L}\log Q(X_l) \\ &= \operatorname*{argmax}_{Q\in\mathcal{Q}} \lim_{L\to\infty} \sum_{l=1}^{L} \log Q(X_l) \end{aligned} \tag{2}$$

Here for the sake of keeping the equations easy to follow, the input alphabet is denoted by $\mathcal{X}$ to mean words in the vocabulary of our language. The distributions under consideration are essentially conditional probability masses conditioned on the same lexical context. Note that Eq. (2) is essentially a maximum likelihood solution. Thus minimization of KL divergence boils down to finding maximum likelihood solution on the simulated corpus. The ML solution can be found out using *n*-gram models.

In the context of our problem, in the limit, as the sampled text size increases to countably infinite, any distribution, $\widehat{Q}^*$, maximizing the likelihood of this sampled text will have 0 KL Divergence with P, provided this distribution also predicts the next word given all the previous context,[7] i.e. that:

$$\lim_{n\to\infty}\lim_{L\to\infty} KL(P\|\widehat{Q}^*) = 0, \tag{3}$$

[5] While implementing this idea practically, we chop the word segments at the end of every end-of-sentence marker – </s>. Note that since </s> is treated as a regular word in our model (i.e. RNN-LMs) and it gets generated which in turn allows the generative model to generate new words when also conditioned on this word token. This, however, may not be true for other long span models and hence for such other models it might be necessary to re-start the generation process afresh from the start of the sentence token – <s>.

[6] Which is true for language models.

[7] We are assuming that we have sampled sufficiently enough times for Maximum Likelihood estimate to be as close to the underlying *true* (but known) distribution as possible.

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

where L is the size of the simulated corpus and n the order of the Markov approximation.

In practice, however, we choose n such that first pass decoding is tractable; L is chosen as large as possible, subject to memory constraints and diminishing returns in LVCSR performance. Since L is finite, for pragmatic purposes, smoothing of n-gram probability distribution (for modeling Q) allows us to approximate the maximum likelihood probability had we seen an infinite corpus ($L \rightarrow \infty$) generated by P, thus we obtain the solution using the following approximation of Eq. (2):

$$\widehat{Q}^* \approx \underset{Q \in \mathcal{Q}}{\operatorname{argmax}} \sum_{l=1}^{L} \log Q(X_l) \tag{4}$$

where $L < K$ and $\mathcal{Q}$ is chosen to be some finite order n-gram family and K is some threshold deciding the size of the corpus to be simulated.

But now the question is whether among all the finite order n-gram models with fixed n, the maximum likelihood finite order n-gram model estimated from the simulated corpus is closest to the generating model? The answer is yes and it is clear from Eq. (2) when the family of distributions, $\mathcal{Q}$, is restricted to be of order n.

In any case we do not know the true distribution generating the true text. However, with our approach, we can at least come close to the generative model of our choice (since we now have control on the amount of data generated under the model of our choice). If that generative model happens to be better than all the other competing generative models, then our approximated model would best imitate it and hopefully be better than the n-gram LM built from the training data using ML principles.

3. Long-span language models

It is well known that humans can exploit longer context with great success in guessing the next word in a sentence (Shannon, 1951). It seems natural therefore to construct LMs that implicitly capture temporal information of arbitrary length. There have been many successful attempts at capturing such long-span information in a language model. Some of these LMs that have shown a great promise in capturing long span information for better prediction are: (a) structured LM of Chelba and Jelinek (2000) which conditions the prediction of next word not only on the previous few lexical tokens, but also on the exposed head words as given by the dependency parse tree of the incomplete sentence under consideration. Such a technique is useful in capturing long-distance dependencies in a natural language; (b) decision Tree based LM of Xu (2005) which asks series of questions of the history for better clustering of the context. Such clustering of the history fights data sparsity problem when the length of the context is increased, thus allowing the model to ask question about distant past for better prediction; (c) recurrent neural network LM (RNN-LM) of Mikolov et al. (2010) which projects the entire context into some lower dimensional space and implicitly clusters "similar" histories for better prediction especially in instances when the lexical pattern (history followed by the predicted word) is not observed during training time. Interested readers can find detailed comparison of advanced language modeling techniques in (Mikolov et al., 2011a).

All of the above, and many other complex LMs, aim to capture long-span information. However, the quality of the model comes at a price: computational complexity in the decoder of speech recognition. Our goal in this paper is to demonstrate that using the proposed variational approximation technique, we can not only incorporate such long-span information available through these models but also make the decoding problem a tractable one.

To illustrate our technique, we have chosen recurrent neural network LM as the candidate long-span LM. It should be, however, noted that our technique is general enough and can be applied to other complex long span language models. Below, we describe in brief the model structure of RNN-LM

3.1. A recurrent neural net language model

In our recent work with a RNN-LM (Mikolov et al., 2010, 2011a,b,c), we showed remarkable improvements in perplexity over n-gram LMs, along with improvement in recognition accuracy. RNN-LMs were also shown to outperform some combinations of syntactic and n-gram models (see Deoras et al., 2010; Mikolov et al., 2011a). We therefore have chosen RNN-LM as our candidate long-span model, which we will try to approximate via n-grams using the proposed technique.

The network has an input layer x, a hidden layer s (also called state or context layer) and an output layer y. Input to the network at time t is denoted $x(t)$, output $y(t)$, and the hidden state $s(t)$. The input $x(t)$ is formed by concatenating a vector $w(t)$, which represents the current word, with output from the context layer $s(t-1)$ to capture long-span dependencies. Fig. 2 shows the schematic representation of such a network.

$$\mathbf{x}(t) = \left[\mathbf{w}(t)^T \mathbf{s}(t-1)^T\right]^T \tag{5}$$

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right) \tag{6}$$

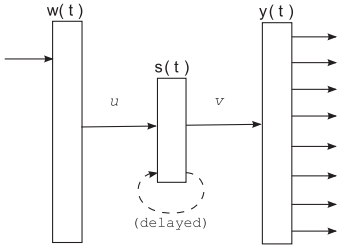

Fig. 2. Schematic representation of recurrent neural network language model. The network has an input layer **w**, a hidden layer **s** and an output layer **y**. Matrices U and V represent synapses.

$$y_k(t) = g\left(\sum_j s_j(t) v_{kj}\right) \tag{7}$$

where $f(z)$ is a sigmoid activation function:

$$f(z) = \frac{1}{1+e^{-z}} \tag{8}$$

and $g(z_m)$ is a softmax function (to make sure that the outputs form a valid probability distribution, i.e. all outputs are greater than 0 and their sum is 1):

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{9}$$

The recurrent architecture of the neural network based language model can be trained by the back-propagation algorithms (BP). However, for better performance, the so-called back-propagation through time (BPTT) algorithm can be used to train the recurrent architectures. With simple BP, the recurrent model can perform poorly in some cases. The BPTT algorithm is also described in (Rumelhart et al., 1986), and a good description for a practical implementation is in (Boden, 2002). The training algorithm is on-line gradient descent, thus the weights are updated after propagating every example. The cross entropy criterion is used to obtain an error vector in the output layer, which is then back propagated to the hidden layer etc. The training algorithm uses validation data for early stopping and to control the learning rate. Training iterates over all the training data in several epochs before convergence is achieved – usually 8–20 epochs are needed. We refer the interested readers to (Mikolov et al., 2010; Mikolov et al., 2011b,c) for more details.

4. Experiments, results and discussion

We report experimental results on four corpora. Perplexity measurements on the Penn WSJ Tree-Bank show that a variational 5-gram is competitive with the best reported results for syntactic LMs. Word error rate (WER) reduction in adapting a Broadcast News language model to the MIT Lectures data is shown next. WER reductions are demonstrated on the NIST 2007 Meeting Recognition (rt07s) and the NIST 2001 Conversational Telephone Recognition (eval01) test sets in the following experimental setup. Finally, improvements are also shown on a very competitive setup of Broadcast News – rt04.

4.1. Perplexity experiments on WSJ

We trained n-gram and RNN-LMs on Sections 0–20 (1M words) of the Penn Tree-Bank corpus, and measured their perplexity on Sections 23 and 24 (0.1M words). Sections 21 and 22 were used as a held out set for parameter tuning.

Baselines: We used interpolated Kneser–Ney smoothing to build 3-gram and 5-gram LMs (containing 0.86M and 2M unique n-gram entries respectively); we will call them the KN models. We also trained an RNN-LM, which we will call RNN-Full. To obtain an alternative long-span model we also trained a *cache* LM from the same training data.

For all models, the vocabulary comprised the 10K most frequent words in Sections 0–20.

Variational approximations: we *simulate* text using the distribution RNN-Full using the sampling procedure described in Section 2.1.

We sampled about 300M word tokens using RNN-Full as a generative model. From this sampled corpus, we estimated a 3-gram and 5-gram Kneser–Ney smoothed LMs. These LMs contained 35M and 42M unique n-gram entries respectively. We pruned these LMs using entropy-pruning (Stolcke et al., 1998) resulting in models with 4.4M and 5.4M n-grams respectively (see Table 1 for the effect of pruning thresholds on the perplexity). We will call them the VarApxRNN models. Each of these n-gram approximations were also interpolated with the corresponding n-gram LMs estimated from (only) the original LM training corpus; these interpolated LMs will be called the VarApx+KN models.

The first column of Table 2 shows that VarApxRNN performs as well as the KN model of the same n-gram order, and their interpolation, VarApx+KN, outperforms both of them. Since the VarApxRNN model is trained on only the *simulated* text, interpolating it with KN introduces the knowledge present in the original training data (Sections 0–20) bringing the simulated statistics closer to the true distribution. To our knowledge, the perplexity of the RNN-full model is significantly lower than any n-gram model reported in the literature.

Fig. 3 empirically supports the asymptotic validity of (3) in the size L of the simulated corpus and model order n.

Comparison with variational approximation of n-gram model: Before we compare the performance of VarApx models with other state-of-the-art language models, it is important to see how does variational model of a KN smoothed n-gram model perform in comparison to the variational model of RNNs. In order to carry out this investigation, we sampled 300M word tokens from KN (5g) model and estimated a 5-gram KN smoothed language model from this sampled data. We will refer to this model as VarApxNgram model. Fig. 4 shows the performance (in terms of perplexity) of variational model of long span

Table 1
The effect of pruning the VarApxRNN (5g) model on the perplexity on Penn Tree-Bank Sections 23 and 24. Pruning the model with 1*e*-08 threshold reduces the model size seven times increasing the perplexity only slightly.

Pruning thresholds	# n-Grams in LM	PPL
0	42M	139.2
1*e*-08	5.4M	140.0
1*e*-07	0.8M	152.1
1*e*-06	0.15M	186.3
1*e*-05	27K	270.0

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

Table 2
LM perplexity on Penn Tree-Bank Sections 23 and 24. These results suggest that RNN-Full is a good approximation to the true distribution of the WSJ text. As a result, VarApx+KN (5g) does exceedingly well in comparison to more complex models which suffer higher variance due to the limited (1M words) text corpus.

Setup	PPL	Setup	PPL
KN (3g)	148	Random forest (Xu)	132
VarApxRNN (3g)	152	–	–
VarApx+KN (3g)	**124**	–	–
KN (5g)	141	SLM (Chelba)	149
VarApxRNN (5g)	140	SLM (Roark)	137
VarApx+KN (5g)	**120**	SLM (Filimonov)	125
VarApx+KN + Cache	111	X-sent (Momtazi)	118
RNN-Full	102	–	–

RNN-LM versus variational model of *n*-gram language model as a function of the size of their corresponding simulated data. It is interesting to see that although VarApxN-gram model start out with a lower perplexity, with more data, VarApxRNN model outperforms VarApxNgram model by 13 perplexity points.

Table 3 shows the performance of variational approximation of RNN and *n*-gram models with and without interpolation with their corresponding full models. It can be seen that sampling data from *n*-gram models does not add too much complementary information. It is thus interesting to see that although long span models, such as RNNs, are estimated from the same training data, due to their complex architecture compared to *n*-gram models, they learn internal representation of the data very well and hence sampling data from such models result in diversity and building a model on top of it reduces the perplexity to a great extent. This also results in improved recognition performance (we show experiments in the later se ctions to support this claim).

Comparison with other long-span LMs: an advantage of choosing the Penn Tree-Bank corpus and the particular training/test partition is that several other researchers have reported perplexity on this setup using various long-span LMs. The second column of Table 2 collects a few such results.

- The random forest language model (RFLM) of Xu (2005) asks questions of only the tri-gram history, and is therefore comparable with VarApxRNN (3g) and VarApx+KN (3g). The RFLM estimates a better 3-gram model from existing text; by contrast, VarApxRNN performs simple estimation from simulated text. It appears that VarApx+KN (3g), which combines simulation with the original text, is better.
- Structured language models have been proposed by Chelba and Jelinek (2000), Roark (2001) and Filimonov and Harper (2009) to exploit *within sentence* long-span dependencies. Table 2 suggests that they are outperformed by VarApx+KN (5g), i.e. by simulating text with RNN-Full and estimating KN 5-gram models.
- Across-sentence dependencies are exploited in the model of Momtazi et al. (2010). This performance is nearly matched by VarApx+KN (5g), which only uses the 5-gram context. Moreover, the across-sentence model is a complex interpolation of many word and class models with regular and skip *n*-grams. The interpolation of VarApx+KN (5g) with another tractable long-span LM, namely the cache LM, outperforms the across-sentence model.

These results suggest that RNN-Full is actually a good approximation to the true distribution of the WSJ text, and the reduction in variance by simulating 300M words of text offsets the bias of the *n*-gram LM estimated from it. As a result, VarApx+KN (5g) outperforms more sophisticated models that have smaller bias, but suffer higher variance due to the limited (1M words) text corpus.

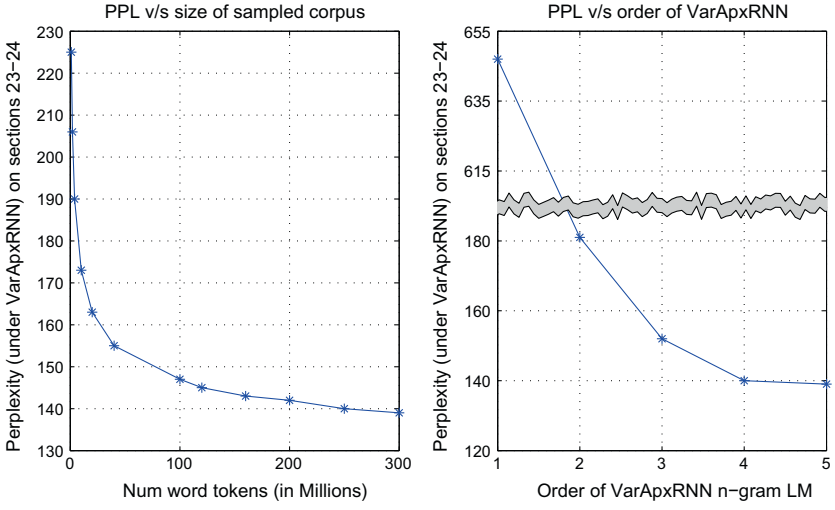

Fig. 3. The perplexity of Sections 23 and 24 as a function of (left) the size L of the simulated corpus for model order $n = 5$ and (right) the order n of the model for corpus size $L = 300$M. These results support (3), but also suggest that VarApxRNN (5g) is still far from the RNN-LM, whose perplexity is 102.

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

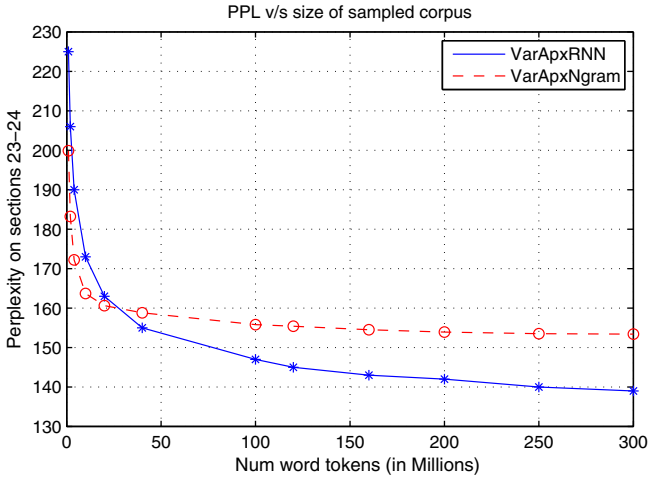

Fig. 4. The perplexity of Sections 23 and 24 computed under VarApxRNN (5g) and VarApxNgram (5g) models, as a function of the size L of the simulated corpus (generated under RNN-Full and KN (5g) respectively).

Table 4 below shows how many novel n-grams were introduced by the variational model. It can be seen that out of a total of 82,430 5-grams that can be extracted from the evaluation section of the Penn Corpus Sections 23 and 24, the variational model needs to back-off for nearly 89% of the 5-grams while the standard baseline model needs to back off for nearly 95% of the 5-grams. The variational model thus creates many new and *relevant* 5-grams which helps in doing a much better prediction of the text data. It can also be seen that the variational model backs-off to relatively higher order models much more frequently than the baseline model. The baseline model backs-off all the way to uni-gram models for about 19% of the 5-grams, while the variational model backs-off to uni-gram model for only about 3.5% of 5-grams. We have observed that most of the speech recognition errors contributed by language model are due to backing-off to lower order models. Lower the order the language model backs-off to, more are the errors induced. The variational model generates novel and relevant n-grams of varying order and reduces the need to back-off to weaker lower order models. This not only helps in reducing the perplexity but also word error rate, as will be seen in the next few experiments.

Table 3
LM perplexity on Penn Tree-Bank Sections 23 and 24 under variational approximation of RNN and Ngram model before and after interpolation with their corresponding full models.

Setup	PPL
KN (5g)	141
VarApxNgram (5g)	153
VarApxNgram+KN (5g)	139
RNN-Full	102
VarApxRNN (5g)	140
VarApx(RNN)+KN (5g)	120

Table 4
Back-off hit ratios for baseline and proposed models (evaluated on a total of 82430 5-grams found in the evaluation set). Richness of variational model can be seen by noting that relatively fewer number of 5-grams in the evaluation section of the Penn Corpus had to back off to lower order models.

# of 5-grams in eval section s.t.	KN (5g) (%)	VarApx+KN (5g) (%)
No backing off	4.90	10.97
Backing off to 4-grams	6.24	15.83
Backing off to 3-grams	19.53	35.97
Backing off to 2-grams	50.41	33.80
Backing off to 1-grams	18.92	3.43

But now the question is whether all of the generated n-grams are relevant? Since the sampling procedure does not filter out *bad* histories, it is likely that the model may produce non-sensical n-grams. In order to test how many n-grams are really relevant, we scour the world wide web (WWW) data. Although a language model built from the web may not be a good one (mainly due to poor normalization across domains and genres), the web will at least be a good corpus to test if a sub sequence appeared anywhere or not. Using innovative web scouring tools developed by Google and Hopkins scientists (Lin et al., 2010), we take each n-gram as produced by variational model and check if such a pattern exists in the web. However, before we carry out such a study, it becomes necessary to establish a baseline. We hence take valid n-grams from the language model trained on the original training data (Sections 0–20) and scour the web for each of them. Table 5 below shows the coverage of n-grams when checked against the web data. The n-grams that were used to query the web were filtered of those n-grams which carried <unk>, <s> and </s>. The n-grams of the variational model were further filtered of those n-grams which were present in the baseline model. Thus the n-grams corresponding to the variational model were purely *synthetic*. We used about 5M 5-grams from the full blown variational model for this study.

From Table 5 we can see that the variational model not only generates many novel n-grams but about 70% of them are in fact found in the web implying that these n-grams are not some random non-sensical patterns. Eighty-five percent of the regular n-grams (as seen in the original training data) are seen in the web implying that the remaining 15% of the n-grams are corpus specific. It is hence reasonable to assume that out of the remaining 28% of the synthetic

Table 5
Coverage of n-grams as checked against the web data World Wide Web (WWW). It can be seen that around 72% of the n-grams produced by the variational model are found on the web.

Model	# of n-grams queried	# n-grams found in web	Coverage (%)
KN (5g)	50,073	42,824	85.5
VarApx (5g)	5,779,012	4,160,889	72

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

n-grams, many n-grams may still be comprehensible and reasonable but probably too corpus specific.

4.2. Domain adaptation experiments on MIT lectures

We performed recognition on the MIT lectures corpus (Glass et al., 2007) using state-of-the-art acoustic models trained on the English Broadcast News (BN) corpus (430 hours of audio), provided to us by IBM (Chen et al., 2009). IBM also provided us its state-of-the-art speech recognizer, Attila (Soltau et al., 2010) and two LMs containing 54M and 4.7M n-grams ($n \leqslant 4$) that were trained on BN text (400M word tokens). Another 150K words of MIT lecture transcripts were provided as in-domain text.

Interpolated Kneser–Ney smoothed n-gram models built with the 150K word in-domain corpus were interpolated with corresponding n-gram LMs from IBM. We, however, used only 4.7M n-gram LM for interpolation as 54M n-gram LM did not provide us any extra benefit. We will call this interpolated model as KN:MIT+BN (interpolation of MIT LM and IBM's 4.7M n-gram LM.). The order of the language model used in first pass decoding was 3, while the order of the LM used for lattice re-scoring was 4. The tri-gram KN:MIT+BN language model contained 4M unique n-grams while 4-gram KN:MIT+BN model contained 4.8M unique n-gram entries. We did not prune the in-domain language model and hence the size of the models were the biggest we could obtain from the available training data. The RNN-LM trained on the 150K words (only) will be called RNN-Full as before. We set the size of the hidden layer to 100.

We simulated text (30M word tokens) using RNN-Full, and estimated n-gram LM from it, which we will again call VarApxRNN. Models resulting from the interpolation of VarApxRNN and KN:MIT+BN n-gram LMs of the same order will be called VarApx+KN. We pruned VarApxRNN model using entropy-pruning (Stolcke et al., 1998) so as to be able to use it in first pass decoder and lattice re-scoring. Thus the tri-gram VarApxRNN language model contained 4.4M unique n-grams while 4-gram VarApxRNN model contained 8.0M unique n-gram entries.

It should be noted that although we synthesized nearly 30 times more data than what the original training corpus already had, the size of the variational model was nearly the same as that of its maximum likelihood model counterpart, at-least for the first pass.

We followed IBM's multi-pass decoding recipe (Soltau et al., 2010) using 3- and 4-gram LMs in the first pass, generated word lattice and N-best list, and rescored them with bigger n-gram models and RNN-Full. Table 6 shows the WER for different decoding configurations, contrasting standard n-gram LMs with the corresponding VarApx+KN LMs. Note that since our re-scoring model was a long span model, it was infeasible to carry out *exact* lattice re-scoring and hence instead we extracted N best lists. We also made use of Iterative Decoding techniques (Deoras et al., 2011a) for re-scoring word lattices directly, whenever N-best list decoding yielded sub-optimal results.

Table 6
Performance (%WER) on the MIT Lectures data set. Decoding with VarApx+KN consistently produces lattices with lower *oracle* WER compared to lattices produced by standard n-gram models. The 1-best output with VarApx+KN also is better than its standard n-gram counterpart.

Setup	Set 1	Set 2
KN:MIT+BN (3g) decoding	24.8	22.4
+KN:MIT+BN (4g) lattice re-scoring	24.8	22.4
+RNN-Full re-scoring (**100** best)	24.1	22.4
+RNN-Full re-scoring (**2000** best)	23.8	21.6
+VarApx+KN (4g) lattice re-scoring	24.2	21.7
Oracle (**2000** best)	17.9	15.5
VarApx+KN (3g) decoding	24.4	22.2
+VarApx+KN (4g) lattice re-scoring	24.1	21.7
+RNN-Full re-scoring (**100** best)	23.8	21.7
+RNN-Full re-scoring (**2000** best)	23.6	21.5
Oracle (**2000** best)	17.5	15.1

We used two sets for decoding. The audio for each set was about 2.1 hours long.

From Table 6, we can see that decoding with VarApx+KN consistently produces lattices with lower *oracle* WER compared to lattices produced by standard n-gram models. The 1-best output with VarApx+KN also is better than its standard n-gram counterpart. In order to see the effect of variational models in isolation i.e. without interpolating them with their maximum likelihood n-gram counterparts, we took VarApxRNN model and interpolated them with just BN models i.e. out-of-domain model – IBM's 4.7M n-gram language model. We will refer to this model as VarApx+KN(-MIT) (3g) and VarApx+KN(-MIT) (4g). '-MIT' signifies that these models were *not* interpolated with n-gram LMs built from in-domain MIT lecture data. Thus these models capture information available only in the form of in-domain simulated data and out-of-domain BN data. The original in-domain training data is ignored completely. Table 7 compares and contrasts performance under various configuration of language models. If we look at 4-gram models, we can see that variational models interpolated with BN models but without MIT n-gram counterparts – VarApx+KN(-MIT) (4g), perform slightly worse than interpolation of MIT and BN language models – KN:MIT+BN (4g) (24.8% → 25.0%) suggesting

Table 7
Performance (%WER) on the MIT Lectures data set when different models are used directly in the first pass decoding or for lattice re-scoring.

Setup	Set 1	Set 2
KN:MIT+BN (3g) decoding	24.8	22.4
+KN:MIT+BN (4g) lattice re-scoring	24.8	22.4
VarApx+KN (3g) decoding	24.4	22.2
+VarApx+KN (4g) lattice re-scoring	24.1	21.7
VarApx+KN(-MIT) (3g) decoding	25.1	22.5
+VarApx+KN(-MIT) (4g) lattice re-scoring	25.0	22.3

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

that simulated data has nearly the same information content as the original training data, however, superior performance of VarApx+KN (4g) model (24.1%) suggests that the information is complementary and hence interpolating model built on original training data and simulated data improves the performance of speech recognition system.

In the perplexity based experiments we saw that many synthetic *n*-grams are actually seen elsewhere in web implying that quite a lot of them are comprehensible and reasonable. From the speech experiments we can see that an increased coverage also brings down the word error rate. For the first set of the above setup, we found out the *n*-gram instances in the reference transcripts such that the *n*th word was either substituted by some other word or was deleted during the automatic transcription. For instance in the toy reference and aligned hypothesis shown below, we can see that the word `for` was erroneously substituted by the word `four` and the word `us` got deleted. We do not consider false insertions in hypotheses because this information cannot be obtained from reference transcripts alone and in this analysis we care only about clean *n*-grams in the reference transcripts such that the predicted word does not get transcribed correctly.

```
REF:    today is a holiday    for     us
HYP:    today is a day        four    **
```

For each such word in the reference transcript, we extracted the *n*-grams (tri-grams corresponding to the words `for` and `us` will be `a holiday for` and `holiday for us`). We then found the coverage of such *n*-grams in the language model. For the hypotheses generated under the baseline models i.e. KN:MIT+BN (4g) model, Table 8 summarizes the findings:

From Table 8 we observe that out of 3204 total *n*-grams such that the *n*th word is falsely recognized by the model using KN:MIT+BN (4g) models, only 19.81% of these were found in this LM. As against that, if we search these *n*-grams in the VarApx+KN (4g) model, we see that 34.86% of them are found. While an improved coverage does not necessarily mean that the WER will be reduced too,[8] but at-least the variational model is able to predict these *n*-grams while the baseline model does not even have them and a prediction for such *n*-grams would require them to backoff to poor lower order models.

Similarly, we extract the *n*-grams from the reference with respect to the hypotheses generated under the variational model. Table 9 summarizes the findings.

Although the coverage in percentage seems to be the same for both Tables 8 and 9, it should be noted that there are fewer errors in the output of the variational model and the total number of *n*-grams seen in the variational model is also small in the latter case. This may imply that an improved coverage of *n*-grams influences reduction of WER to a large extent.

[8] The LM may have an entry for a particular *n*-gram, but if it is assigned a very low probability, then probably the recognizer wont be able to produce it.

Table 8
Coverage of falsely recognized (wrt baseline model) *n*-grams as checked against the language models. It can be seen that the baseline model has a very poor coverage while the variational model has nearly double coverage. Increasing the coverage is likely to improve the performance of the speech recognition system.

Model	# of *n*-grams queried	# *n*-grams found in LM	Coverage (%)
KN:MIT+BN (4g)	3204	635	19.81
VarApx+KN (4g)	3204	1117	34.86

Table 9
Coverage of falsely recognized (wrt. variational model) *n*-grams as checked against the language models. The absolute number of *n*-grams covered by the language models are less than that in Table 8 implying that an improved coverage of *n*-grams influences reduction of WER to a large extent.

Model	# of *n*-grams queried	# *n*-grams found in LM	Coverage (%)
KN:MIT+BN (4g)	3141	600	19.10
VarApx+KN (4g)	3141	1067	33.97

4.3. Conversational speech recognition experiments

We demonstrate WER improvements on two conversational speech recognition tasks: the transcription of multi party meetings, and of conversational telephone speech (rt05s, rt07s respectively). Two system setups were used for meeting and telephone speech data respectively. Brno's variant of the AMI system, developed for the NIST Meeting Transcription evaluation (Hain, 2005), was used for the former, and the Brno conversational telephone speech (CTS) system for the latter.

The AMI recognizer used fast speaker adaptation (HLDA, CMLLR and VTLN); it processed PLP+NN-posterior features extracted from 16 kHz audio with SAT models trained on 200 hours of meeting data. The CTS recognizer used an initial decoding pass for VTLN and MLLR, and processed PLP features extracted from 8 kHz audio with SAT models trained on 270 hours of telephone speech. All acoustic models were trained using the MPE criterion and used cross-word tied-state tri-phones, and both setups produced bi-gram lattices using a 2-gram LM trained using Good-Turing discounting, which were subsequently expanded to 5-gram lattices using a modified Kneser–Ney smoothed LM.

5M words of Fisher CTS transcripts were used as training text for three LMs: two *n*-grams and an RNN. We call the 2-gram model with Good-Turing discounting GT (2g). GT (2g) contained 1M unique *n*-gram entries. The 5-gram model and the RNN model are called KN (5g) and

Table 10
Performance (%WER) on conversational speech data sets. VarApx+KN reduces the WER by 0.7–0.8% over a 5-gram model on both telephone speech (eval01) and meetings (rt07s).

Setup	eval01	rt07s
GT (2g) decoding	30.3	33.7
+KN (5g) lattice re-scoring	**28.0**	**32.4**
+RNN-Full re-scoring (**100** best)	27.1	30.8
+RNN-Full re-scoring (**1000** best)	26.5	30.5
+VarApx+KN (5g) lattice re-scoring	27.3	31.9
Oracle (**1000** best)	19.5	21.3
VarApx+GT (2g) decoding	30.1	33.3
+VarApx+KN (5g) lattice re-scoring	**27.2**	**31.7**
+RNN-Full re-scoring (**100** best)	27.0	30.6
+RNN-Full re-scoring (**1000** best)	26.5	30.4
Oracle (**1000** best)	19.5	21.0

RNN-Full, as before. KN (5g) model contained 3M unique n-gram entries. 400M words of text generated from RNN-Full LM via Monte Carlo sampling were used to estimate additional n-gram LMs, which we again call VarApxRNN. The VarApxRNN after interpolation with their full n-gram counterpart will be called VarApx+GT (2g) and VarApx+KN (5g) model respectively. We pruned VarApxRNN model before using them in any recognition experiment. Thus VarApxRNN (2g) and VarApxRNN (5g) model contained 2.6M and 9M unique n-gram entries respectively. Since the LMs were trained on CTS transcripts, they are in-domain for conversation telephone recognition (eval01), but out-of-domain for meeting recognition (rt07s).

The four LMs were applied to rt07s and eval01, and the WERs are reported in Table 10. The table also illustrates the WER when N-best list re-scoring with RNN-Full is performed.

From Table 10, we can again see that the variational models reduce the WER significantly over the conventional n-gram models. Re-scoring of lattices (output by conventional first pass n-gram models), with variational models achieve similar performance improvements, further suggesting that variational models can also be used for lattice re-scoring stages instead of first pass decoding. Such a setup is favorable in any real world setting where changing the decoder parameters and models is very costly.

4.4. Broadcast news speech recognition experiments

An interesting question that lingers in mind is what happens when the original training data itself is huge. By huge we mean number of word tokens in hundreds of millions. To answer this question, we tested the proposed technique on a very competitive setup of Broadcast news. The setup details and results are presented next:

We performed recognition on the Broadcast News (BN) `rt04` task using state-of-the-art acoustic models trained on the English Broadcast News (BN) corpus (430 hours of audio) provided to us by IBM (Chen et al., 2009). IBM also provided us its state-of-the-art speech recognizer, Attila (Soltau et al., 2010) and two Kneser–Ney smoothed back-off n-gram LMs containing 4.7M n-grams ($n \leqslant 4$) and 54M n-grams ($n \leqslant 4$) were trained. We will refer to them as KN:BN-Small and KN:BN-Big respectively. The LM training text consists of 400M words from the following data sources: 1996 CSR Hub4 Language Model data, EARS BN03 closed captions, GALE Phase 2 Distillation GNG Evaluation Supplemental Multilingual data, Hub4 acoustic model training transcripts, TDT4 closed captions, TDT4 newswire, and GALE Broadcast Conversations and GALE Broadcast News.

We trained an RNN based language models, denoted further as RNN-Full, on the entire broadcast news training data (400M word tokens). We used interpolation of 3 RNNs with 320, 480 and 640 neurons in the hidden layer respectively.

From the generative model – RNN-Full, we generated data of varying sizes containing 160M, 500M, 1B and 2.5B word tokens. We estimated Good-Turing (GT) smoothed 4-gram language models referred to as VarApx-160M, VarApx-500M, VarApx-1B and VarApx-2.5B respectively.[9] As we will show later, its our observation that models built on bigger data sets and then later pruned result in better performance than similar sized full blown models built, however, on small data sets. We hence use VarApx-2.5B with varying pruning and use it either for first pass or re-scoring with and without interpolating it with KN:BN-Big, depending upon the experimental setup. Detailed information about various variational models and the effect of size of simulated data and pruning of models on the performance is reported in Tables 12–14. Interpolation of VarApx-2.5B, containing 50M n-grams (obtained after pruning with 1e-09 threshold) and KN:BN-Big will be referred to as VarApx+KN-Big. We pruned VarApx-2.5B further to bring it to the size roughly equivalent to KN:BN-Small. We, however, do not interpolate it and this model becomes our choice for the first pass decoding. We will refer to this model simply by VarApx-Small. This model contained as little as 3.5M n-grams. VarApx-Small is thus comparable to KN:BN-Small in terms of size. The interpolation of VarApx-Small and KN:BN-Small will be referred as VarApx+KN-Small.

Following table summarizes the WER of various models on the evaluation set `rt04`.

From Table 11, we can see that VarApx-Small reduces the WER by 0.5% absolute over a 4-gram model when used directly in the first pass decoding. VarApx+KN-Small decoding reduces the WER by 0.8% absolute over KN-BN:Small decoding. Re-scoring lattices, output by either methods, with VarApx+KN-Big improves the performance further. These numbers are very encouraging as this

[9] We found that Good-Turing smoothing works better for models estimated from large amount of sampled data than modified Kneser–Ney smoothing.

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

Table 11
Performance (%WER) on Broadcast News speech data set (rt04). VarApx-Small reduces the WER by 0.5% absolute over a regular 4-gram model when used directly in the first pass decoding. Re-scoring lattices, output by either methods, with VarApx+KN-Big improves the performance further. VarApx+KN-Small improves the performance over KN:BN-Small decoding by 0.8% absolute.

Setup	rt04
KN:BN-Small decoding	**14.1**
+KN:BN-Big lattice re-scoring	13.1
+VarApx+KN-Big lattice re-scoring	12.8
+RNN-Full re-scoring	11.7
VarApx-Small decoding	**13.6**
+VarApx+KN-Big lattice re-scoring	12.8
+RNN-Full re-scoring	11.7
VarApx+KN-Small decoding	**13.3**

setup is very competitive and the original training data itself is huge. So the performance improvements over the state-of-the-art *n*-gram language model is very interesting.

It is interesting to see how the performance of variational models change when they are built on data of gradually increasing size. Table 12 shows that the performance of the variational model with and without KN:BN-Big interpolation, becomes better when more sampled data is used to train these models. In this table, we consistently used 1e-10 as the threshold to prune the full blown models, eventually resulting in the number of unique *n*-grams as reported. Table 13 shows the performance of the models when trained on the original BN training data.

The immediate conclusion from Table 12 is that with more sampled data, the performance of the variational model and in combination with KN:BN-Big improves the performance leading to a 0.3% absolute WER reduction over that of re-scoring with KN:BN-Big model. Variational model by itself outperforms KN:BN-Big model by 0.1% absolute WER.

Usually the size of the *n*-gram model is correlated with the size of the data from which it is estimated. Hence with more sampled data, the size of the variational model also increases. This can potentially be a bottleneck for first pass decoding and even for re-scoring the lattices. We hence investigated the effect of pruning the models on the performance (both, perplexity and word error rate). We take 2.5B word token sampled corpus and build models with varying pruning thresholds. Table 14 shows that even with aggressive pruning, the model performance does not degrade much. On the other hand, the size of the model reduces considerably with pruning. The variational model containing 50M *n*-grams is thus comparable to KN:BN-Big in terms of size. By itself, this model outperforms KN:BN-Big. Interpolation with KN:BN-Big reduces the WER by 0.3% absolute, which is significant.

Table 12
Perplexity and re-scoring results on dev04f and rt04 data respectively as a function of the size of the sampled data. Models trained on the sampled data (always pruned with 1*e*-10 threshold) is evaluated with and without interpolating with KN:BN-Big (referred to as BN-Big in the table to save space). KN:BN-Small was used in first pass to produce lattices. With more sampled data, the performance of individual model and in combination with KN:BN-Big improves the performance leading to a 0.3% absolute WER reduction over that of re-scoring with KN:BN-Big model.

# words/# *n*-grams	PPL		WER	
	w/o BN-Big	w/ BN-Big	w/o BN-Big	w/ BN-Big
160M/33M	189.7	137	13.52	13.03
500M/89M	175	134.4	13.35	13.00
1B/109M	164	132.3	13.21	12.93
2.5B/223M	159.4	130.1	13.03	12.82

Table 13
Perplexity and WER on dev04f and rt04 data respectively for the models: KN:BN-Small and KN:BN-Big trained on original BN training data.

# words/# *n*-grams	PPL	WER
(KN:BN-Small) 400M/4.7M	180.3	14.14
(KN:BN-Big) 400M/54M	144	13.11

Table 14
Perplexity and re-scoring results on dev04f and rt04 data respectively as a function of the size of the model estimated from 2.5B word token sampled corpus. Pruning the model with 1*e*-09 threshold, makes the model considerably smaller with respect to the model obtained with 1*e*-10 pruning, but resulting in almost same performance. Such models become feasible to be used in any practical speech recognition setups.

# *n*-grams/# pruning thresholds	PPL		WER	
	w/o BN-Big	w/ BN-Big	w/o BN-Big	w/ BN-Big
223M/1*e*-10	159.4	130.1	13.03	12.82
50M/1*e*-09	159.6	130.9	13.04	12.84
8M/1*e*-08	173.6	135.2	13.36	12.94
1.4M/1*e*-07	209.3	141.6	13.90	13.10

4.5. Analysis and discussion of LVCSR results

From Table 6 it is clear that using VarApx+KN during decoding consistently produces lattices with a 0.5% lower *oracle* WER compared to lattices produced by standard *n*-gram models. The first pass output from decoding with VarApx+KN also has 0.2–0.4% lower WER than from decoding with their standard *n*-gram counterparts. However re-scoring with language models with 4-gram order produces even better results. We can see 0.7% absolute reduction after the application of 4-gram LMs on the lattices produced by the corresponding 3-gram LMs. It seems fair to conclude that VarApx+KN is a better *n*-gram model than a standard *n*-gram model estimated with Kneser–Ney smoothing. Unlike RNN-Full, it can be incorporated into the decoder, bringing some of the benefits of RNN-Full to first pass decoding. The complementarity of the VarApx model to KN models is further evident from the fact that re-scoring the lattices (produced under standard KN smoothed models) with VarApx+KN model reduces the

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

WER by as much as 0.6% (24.8% → 24.2% and 22.4% → 21.7%).

Note further from the upper half of Table 6 that 2000-best re-scoring with RNN-Full reduces WER over a standard 3-gram by 0.8%–0.9%. In the lower half, using VarApx+KN in decoding shows a different benefit: if VarApx+KN is used for generating the *N*-best list, the same WER reduction is available at 100-best re-scoring. If 2000-best re-scoring is undertaken, an additional small gain of 0.2% is obtained.

Fig. 5 shows that even when the search space size is increased beyond 2000 best, there still remains the gap between the oracle WER of the baseline approach i.e. KN:MIT+BN model and the proposed approach i.e. VarApx+KN model. These experiments show that variational model produces richer search spaces so that even a sub-optimal search (via *N*-best) produces much better results during re-scoring with a full blown model.

The benefits of decoding and lattice re-scoring with the variational approximation of RNN-Full are even more evident from Table 10, where VarApx+KN reduces WER by 0.7–0.8% over a 5-gram on both CTS and meeting transcription (28.0% → 27.2% and 32.4% → 31.7% respectively). Similar to results on MIT corpus, here too, we can see that re-scoring lattices produced under KN smoothed models with VarApx+KN model reduces the WER by as much as 0.7% for eval01 and 0.5% for rt07s (28.0% → 27.3% and 32.4% → 31.9% respectively) further validating the complementarity of the VarApx models.

A final observation from Table 10 is that there still remains a gap between decoding with VarApx+KN and re-scoring with RNN-Full. The latter reduces WER by almost 2% (absolute) over the standard 5-gram, compared to 0.7–0.8% by the former. This suggests that when the RNN is trained on more data (5M words in Table 10 vs. 1M words in Table 2), it improves even further over a 4- or 5-gram model.

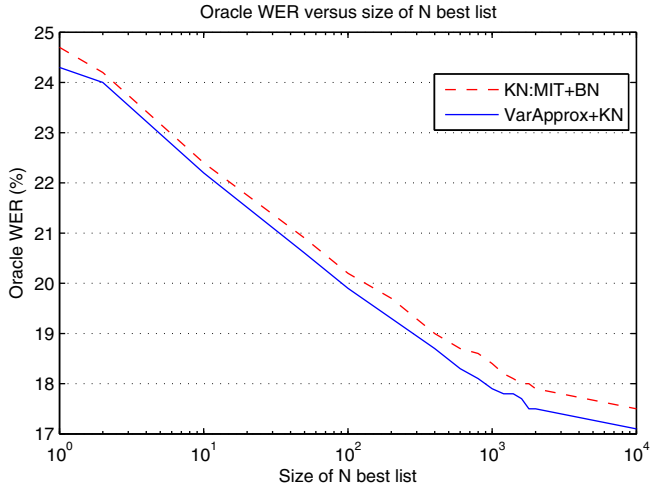

Fig. 5. Plot of Oracle WER versus the size of the search space in terms of number of hypotheses in *N*-best list. It can be seen that although the Oracle accuracy increases with an increase in the search space, there still remains a gap in the performance of the baseline and proposed approach, with the latter consistently better.

Results on larger data-set are very encouraging too and are in line with the conclusions and intuitions from our previous experiments. Table 11 shows the improvements in performance when VarApx-Small is used directly in the first pass decoding. VarApx-Small reduces the WER by 0.5% absolute over a 4-gram model - KN:BN-Small when used directly in the first pass decoding. Interpolation of VarApx-Small and KN:BN-Small in the first pass decoding reduces the WER by 0.8% absolute over KN:B-Small decoding. Re-scoring lattices, output by either models (VarApx-Small or KN:BN-Small), with VarApx+KN-Big improves the performance by 0.3% absolute over KN:BN-Big re-scoring. Previously we saw that for moderately sized training corpora, in order to obtain competitive variational models, we had to generate sometimes as much as 30 times more data. However, an interesting observation from the large data BN experiment is that if the original training data is huge to begin with, the generative model generalizes the data so well that simulating even three or four times synthetic data suffices. Table 12 shows performance of the model as a function of the size of the simulated data. The size of the *n*-gram models is correlated closely with the size of the data from which they are built. However, we observed that aggressively pruning the variational models, not only bring down the models to a more manageable and realistic sizes but also result in very little degradation in performance. See Table 14 for the performance of variational models as a function of the size of the models. We are thus able to create comparable sized models, sometimes two or three times smaller (comparable to models built on original training data) and result in better performance. See Tables 13 and 14 to compare the performance of the regular and variational model as a size of their corresponding training data.

All our *n*-gram decoding results, in all the setups used, are statistically significant (with a $p \leqslant 0.001$). We used NIST's *sclite* toolkit for computing statistical significance using MAPSSWE significance tests.

Given the strong positive improvements using variational models, either for first pass decoding or for lattice re-scoring, we propose the following two recipes for decoding and re-scoring:

1. First pass decoding
   (a) Build as good a complex & long span language model as possible.
   (b) Sample as much text data as possible from this 'generative' model.
   (c) Build *n*-gram model based on the sampled data.
   (d) Prune it down to some reasonable size.
   (e) Interpolate it with the baseline model i.e. *n*-gram model estimated on training data.
   (f) Do first pass decoding with the resulting model.

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

2. Re-scoring word lattices
   (a) Carry out steps 1(a), 1(b) and 1(c) as above.
   (b) Interpolate the *n*-gram based on sampled data with the *n*-gram model estimated from training data.
   (c) Produce lattices with baseline *n*-gram model or with the model in step 1(e) above and rescore them with the model obtained in step 2(b) above.

5. Conclusion

In this paper, we have shown that certain long span and complex language models can be successfully approximated by simple *n*-gram models. In our work, we approximated a near Turing machine model – recurrent neural network based language model. Our method allows easy integration of such complex models into speech recognition or machine translation systems. We have validated our claim on various speech recognition data-sets of varying complexities. We conclude from our experiments that with large amounts of simulated data, the variational model becomes better, resulting in improved performance over conventional *n*-gram models estimated from the original training data. More importantly, pruning such models result in more manageable and realistic *n*-gram models without any significant loss of accuracy or performance. The variational models, by themselves, are good substitutes for the regular *n*-gram first pass decoding models.

Another interesting conclusion from our experiments is that variational models are better choices even for re-scoring i.e. we may not even have to touch the first pass decoding at all. Re-scoring lattices (output by regular *n*-gram models) with variational *n*-gram re-scoring models, result in improvements similar to when the first pass uses variational models in conjunction with re-scoring the output lattices with bigger variational models. Thus in situations where it is expensive to change decoder models and parameters, we can use the variational substitutes and achieve improved performance by just re-scoring the first pass lattices.

From the perspective of machine learning (Bengio, 2007), we can say that we approximate deep architectures (such as recurrent neural network model) by a shallow architecture (*n*-gram model). This approximation comes at a cost of additional memory complexity – we basically store huge database of pre-computed results. As deep architectures are increasingly more popular among speech recognition community (Deng, 2011), we hope that our work will provide more insight into capabilities and limitations of *n*-gram models, and possibly provide inspiration for related work even in other ASR fields.

Acknowledgements

HLT-COE Johns Hopkins University partially funded this research. BUT researchers were partly funded by the Technology Agency of the Czech Republic Grant No. TA01011328, and Grant Agency of Czech Republic Project No. 102/08/0707. Frederick Jelinek's contribution is acknowledged towards this work. He would be a co-author if he were available and willing to give his consent. We thank anonymous reviewers for their many helpful comments and suggestions.

Appendix A. Simulations under RNNs and 4-gram LMs

In this appendix, we present some sample simulations from RNN and 4-gram LM. We provided a common prefix to both type of language models and then generated the remaining text. We have used *TYPEWRITERITALICS* font to indicate that the simulations are from RNN and TYPEWRITER REGULAR font to indicate that the simulations are from 4-gram LM. Below we show simulations from the LMs trained on Broadcast News corpus and Penn Tree Bank corpus respectively.

A.1. Broadcast news corpus

<s> IT WAS DOUBTFUL THAT *HE WOULD ALWAYS SAY HE THINKS DOCTOR GUNN COULD BREAK THE DEATH SENTENCE ESPECIALLY FOR THOSE WHO WERE EXECUTED* </s>

<s> IT WAS DOUBTFUL THAT YOU CAN NEWT GINGRICH IS REPORTS A DAILY POUNDS AS SOMEONE ELSE THERE ARE NOT MANY GRAY OF ROSEANNE WE DO NOT HAVE A PIT BULL </s>

<s> SENATOR BIDEN ALSO SAID *THAT MANY ACTION ON CRIME SHE ARGUED WERE ASCENDING TO THEIR RIGHTS SPAWNS* </s>

<s> SENATOR BIDEN ALSO SAID IF THIS IS SEEN IN THIS COUNTRY AFTER THE GOVERNMENT TO SHAKE HIM POCKETS A DIVERSITY RINGING THE WEIGHT IN BELGIUM HOW THAT'S YESTERDAY ONE STRAIGHT VISITING ICELAND RESPONSIBILITY RACES FISH </s>

<s> WHEN IT COMES TO JUVENILE *JUSTICE SYSTEM HIS PRACTICALITY AND THE WAY TO HOLD HIM CAN EH THE REALLY CATCH AND SUSPICION THAT WHEN THE DISTRICT ATTORNEY IS SUBJECT TO A CRIME BILL WE'RE GOING TO GO TO A DISGRACEFUL SECRET FROM THE D.C.* </s>

<s> WHEN IT COMES TO JUVENILE VERY ILLEGIBLE CARRIED THE HAMPTONS THIRTIES APPROVAL FOR A STILL WORKING INDEPENDENTLY </s>

<s> THE U. N. SECRETARY GENERAL *KOFI ANNAN IS ACCUSED OF ENDORSING DENNIS LI FOR ORDER TO TRY TO GIVE THAT THE DOCUMENT TO REINFORCE AND DEMANDING A CEASE FIRE BUT THE ISRAELIS HAVE DECLARED SOME THREE HUNDRED AND TRAVELING EIGHT BILLION DOLLARS THIS YEAR AND THAT COULD REVOLUTIONARY PUTIN* </s>

<s> THE U. N. SECRETARY GENERAL METHOTREXATE HOSPITALS WOULD TYPICALLY INADVERTENTLY

HIT THAT THE OIL FIELDS FOR GARY CONDIT HAM JORDAN WHO IS INVOLVED IN ALL GROUPS </s>

<s> AND I'M SORRY I *NEVER SAW ANYTHING FLOWING TO ME* </s>

<s> AND I'M SORRY I REACHED MY UNCLE'S ANSWER WHETHER MONDRAGON AND LIFE AS SOON AS THEY KIRK LYONS A RESOUNDING HOWEVER AND THEN SOME </s>

A.2. Penn tree bank corpus

<s> IN ADDITION THE CHARGE *FOR WHITES WERE STRONGER IN THE SUSAN SECTOR* </s>

<s> IN ADDITION THE CHARGE <UNK> THEIR IN PRISON AFTER IT COULD TONY MAKER </s>

<s> NONETHELESS MR. BROWN SAID *THEIR OPERATING A RESULT OF SHAREHOLDERS WAS BELOW THE CONTRACTS* </s>

<s> NONETHELESS MR. BROWN SAID NET OF $ N MILLION </s>

<s> BUT RESEARCHERS STARTED TO *MEET THE ACTIONS WILL RANGE AT FLY* </s>

<s> BUT RESEARCHERS STARTED TO BECOME AND NOW THAT THEY O'KICKI UNEXPECTEDLY WIVES WHETHER WORKING CAPITAL THERE HAS BEEN FILIPINO MOST PEOPLE AND INDUSTRIAL PRODUCTION OF LARGE SIZE INCLUDING FINANCING </s>

<s> SOUTH AFRICA 'S SECOND-LARGEST *BUSINESS STAYED FIRMED N AT LARGE CONSECUTIVE FOUR MONTHS AND HAD DROPPED TO* $ *N* </s>

<s> SOUTH AFRICA 'S SECOND-LARGEST PRODUCTS SHELVES SPRING RETINOBLASTOMA DISCOVERY WILL LAUNCH EXPERIENCE TO SEE A WITHOUT <UNK> </s>

<s> ESTIMATED EARNINGS SAID NET *INCOME THE DECLINE IN INCOME ROSE N N TO* $ *N BILLION FROM* $ *N BILLION* </s>

<s> ESTIMATED EARNINGS SAID NET INCOME N.V. NETHERLANDS </s>

References

Allauzen, C., Mohri, M., Roark, B., 2003. Generalized Algorithms for constructing statistical language models. In: Proc. of Association of Computational Linguistics (ACL).

Bengio, Y., LeCun, Y., 2007. Scaling Learning Algorithms Towards AI. MIT Press, Large-Scale Kernel Machines.

Bickel, P.J., Doksum, K.A., 1977. Mathematical Statistics: Basic Ideas and Selected Topics. Holden-Day Inc., Oakland, CA.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Boden, M., 2002. A Guide to Recurrent Neural Networks and Back Propagation. Dallas Project.

Chelba, C., Jelinek, F., 2000. Structured language modeling. Comput. Speech Language 14, 283–332.

Chen, S.F., Mangu, L., Ramabhadran, B., Sarikaya, R., Sethy, A., 2009. Scaling shrinkage-based language models. In: Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 299–304.

Cover, T., Thomas, J., 1991. Elements of Information Theory. John Wiley and Sons Inc., NY.

Deng, L., 2011. An overview of deep-structured learning for information processing. In: Proc. Asian-Pacific Signal and Information Processing – Annual Summit and Conference (APSIPA-ASC).

Deoras, A., 2011. Search and decoding strategies for complex lexical modeling in LVCSR. PhD thesis, The Johns Hopkins University.

Deoras, A., Jelinek, F., 2009. Iterative decoding: A novel re-scoring framework for confusion networks. In: Proc. IEEE Workshop on Automatic Speech Recognition Understanding (ASRU), pp. 282–286.

Deoras, A., Filimonov, D., Harper, M., Jelinek, F., 2010. Model combination for speech recognition using empirical bayes risk minimization. In: Proc. IEEE-Spoken Language Technologies.

Deoras, A., Mikolov, T., Church, K., 2011a. A fast re-scoring technique to capture long distance dependencies. In: Proc. 2011 Conf. on Empirical Methods in Natural Language Processing (EMNLP).

Deoras, A., Mikolov, T., Kombrink, S., Karafiát, M., Khudanpur, S., 2011b. Variational Approximation of Long-Span Language Models for LVCSR, in: Proc. IEEE Internat. Conf. on Acoustics, Speech, and Signal Processing (ICASSP).

Elman, J., 1990. Finding structure in time. Cognit. Sci. 14 (2), 179–211.

Filimonov, D., Harper, M., 2009. A joint language model with fine-grain syntactic tags. In: Proc. 2009 Conf. on Empirical Methods in Natural Language Processing (EMNLP).

Geman, S., Geman, D., 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Machine Intell. 6, 721–741.

Glass, J., Hazen, T., Cyphers, S., Malioutov, I., Huynh, D., Barzilay, R., 2007. Recent progress in MIT spoken lecture processing project. In: Proc. ICSLP-Interspeech.

Hain, T., 2005. The 2005 AMI system for the transcription of speech in meetings. In: Proc. Rich Transcription 2005 Spring Meeting Recognition Evaluation Workshop, UK.

Jordon, M.I., 1998. Learning in Graphical Models. The MIT Press, Cambridge, MA.

Lin, D., Church, K., Ji, H., Sekine, S., Yarowsky, D., Bergsma, S., Patil, K., Pitler, E., Lathbury, R., Rao, V., Dalwani, K., Narsale, S., 2010. New tools for web-scale *n*-grams. In: Chair, N.C.C., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., Tapias, D. (Eds.), Proc. Seventh Conf. on International Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA), Valletta, Malta.

Mikolov, T., Deoras, A., Kombrink, S., Burget, L., Černocký, J.H., 2011a. Empirical Evaluation and Combination of Advanced Language Modeling Techniques, in: Proc. of ICSLP Interspeech.

Mikolov, T., Deoras, A., Povey, D., Burget, L., Černocký, J.H., 2011b. Strategies for training large scale neural network language models. In: Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU).

Mikolov, T., Kombrink, S., Burget, L., Černocký, J.H., Khudanpur, S., 2011c. Extensions of recurrent neural network language model. In: Proc. IEEE Internat. Conf. on Acoustics, Speech, and Signal Processing (ICASSP).

Mikolov, T., Karafiát, M., Burget, L., Černocký, J.H., Khudanpur, S., 2010. Recurrent neural network based language model. In: Proc. ICSLP-Interspeech.

Momtazi, S., Faubel, F., Klakow, D., 2010. Within and across sentence boundary language model. In: Proc. ICSLP-Interspeech.

Nederhof, M.-J., 2005. A general technique to train language models on language models. In: Proc. Association of Computational Linguistics (ACL).

Nederhof, M.-J., Satta, G., 2004. Kullback–Leibler distance between probabilistic context-free grammars and probabilistic finite automata. In: Proc. Association of Computational Linguistics (ACL).

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004

Roark, B., 2001. Probabilistic top-down parsing and language modeling. Comput. Linguist. 27, 249–276.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning internal representations by error propagation. In: Computational Models of Cognition And Perception Series, MIT Press, pp. 318–362.

Shannon, C., 1951. Prediction and entropy of printed English. Bell Syst. Tech. J. 30, 50–64.

Soltau, H., Saon, G., Kingsbury, B., 2010. The IBM Attila speech recognition toolkit. In: Proc. IEEE Workshop on Spoken Language Technology.

Stolcke, A., 1998. Entropy-based pruning of backoff language models. In: Proc. DARPA Broadcast News Transcription and Understanding Workshop, pp. 8–11.

Stolcke, A., Segal, J.,1994. Precise *n*-gram probabilities from stochastic context-free grammars. In:Proc. Association of Computational Linguistics (ACL).

Xu, P., 2005. Random forests and the data sparseness problem in language modeling. PhD thesis, Johns Hopkins University.

Please cite this article in press as: Deoras, A. et al., Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model, Speech Comm. (2012), http://dx.doi.org/10.1016/j.specom.2012.08.004