2012

A Scalable Distributed Syntactic, Semantic and Lexical Language Model

Ming Tan
Wright State University - Main Campus

Wenli Zhou
Wright State University - Main Campus

Lei Zheng
Wright State University - Main Campus, lei.zheng@wright.edu

Shaojun Wang
Wright State University - Main Campus, shaojun.wang@wright.edu

A Scalable Distributed Syntactic, Semantic, and Lexical Language Model

Ming Tan*
Wright State University

Wenli Zhou**
Wright State University

Lei Zheng†
Wright State University

Shaojun Wang‡
Wright State University

This paper presents an attempt at building a large scale distributed composite language model that is formed by seamlessly integrating an n-gram model, a structured language model, and probabilistic latent semantic analysis under a directed Markov random field paradigm to simultaneously account for local word lexical information, mid-range sentence syntactic structure, and long-span document semantic content. The composite language model has been trained by performing a convergent N-best list approximate EM algorithm and a follow-up EM algorithm to improve word prediction power on corpora with up to a billion tokens and stored on a supercomputer. The large scale distributed composite language model gives drastic perplexity reduction over n-grams and achieves significantly better translation quality measured by the Bleu score and "readability" of translations when applied to the task of re-ranking the N-best list from a state-of-the-art parsing-based machine translation system.

1. Introduction

The Markov chain (n-gram) source models, which predict each word on the basis of the previous $n-1$ words, have been the workhorses of state-of-the-art speech recognizers and machine translators that help to resolve acoustic or foreign language ambiguities by placing higher probability on more likely original underlying word strings. Although the Markov chains are efficient at encoding local word interactions, the n-gram model

* Kno.e.sis Center and Department of Computer Science and Engineering, Wright State University, Dayton OH 45435. E-mail: tan.6@wright.edu.
** Kno.e.sis Center and Department of Computer Science and Engineering, Wright State University, Dayton OH 45435. E-mail: zhou.23@wright.edu.
† Kno.e.sis Center, Wright State University, Dayton OH 45435. E-mail: lei.zheng@wright.edu.
‡ Kno.e.sis Center and Department of Computer Science and Engineering, Wright State University, Dayton OH 45435. E-mail: shaojun.wang@wright.edu.

Submission received: 10 October 2010; revised submission received: 17 October 2011; accepted for publication: 16 November 2011.

clearly ignores the rich syntactic and semantic structures that constrain natural languages. Attempting to increase the order of an *n*-gram to capture longer range dependencies in natural language immediately runs into the curse of dimensionality (Bengio et al. 2003). The performance of conventional *n*-gram technology has essentially reached a plateau (Rosenfeld 2000b; Zhang 2008), and it has proven remarkably difficult to improve on *n*-grams (Jelinek 1991; Jelinek and Chelba 1999). Research groups (Och 2005; Zhang, Hildebrand, and Vogel 2006; Brants et al. 2007; Emami, Papineni, and Sorensen 2007) have shown that using an immense distributed computing paradigm, up to 6-grams can be trained on up to billions and trillions of tokens, yielding consistent system improvements because of excellent *n*-gram hit ratios on unseen test data, but Zhang (2008) did not observe much improvement beyond 6-grams. As the machine translation (MT) working groups stated in their final report (Lavie et al. 2006, page 3), "These approaches have resulted in small improvements in MT quality, but have not fundamentally solved the problem. There is a dire need for developing novel approaches to language modeling."

Over the past two decades, more sophisticated models have been developed that outperform *n*-grams; these are mainly the syntactic language models (Della Pietra et al. 1994; Chelba 2000; Chelba and Jelinek 2000; Charniak 2001; Roark 2001; Wang and Harper 2002; Jelinek 2004; Benedí and Sánchez 2005; Van Uytsel and Compernolle 2005) that effectively exploit sentence-level syntactic structure of natural language, and the topic language models (Saul and Pereira 1997; Gildea and Hofmann 1999; Bellegarda 2000; Wallach 2006) that exploit document-level semantic content. Unfortunately, each of these language models only targets some specific, distinct linguistic phenomena (Pereira 2000; Rosenfeld 2000a, 2000b); thus, each captures and exploits different aspects of natural language regularity. A natural question we should ask is whether/how we can construct more complex and powerful but computationally tractable language models by integrating many existing/emerging language model components, with each component focusing on specific linguistic phenomena like syntactic structure, semantic topic, morphology, and pragmatics in complementary, supplementary, and coherent ways (Bellegarda 2001, 2003).

Several techniques for combining language models have been investigated. The most commonly used method is **linear interpolation** (Chen and Goodman 1999; Jelinek and Mercer 1980; Goodman 2001), where each individual model is trained separately and then combined by a weighted linear combination. All of the syntactic structure-based models have used linear interpolation to combine trigrams to achieve further improvement over using their own models alone (Charniak 2001; Chelba and Jelinek 2000; Chelba 2000; Roark 2001). The weights in this case are trained using held-out data. Even though this technique is simple and easy to implement, it does not generally yield very effective combinations (Rosenfeld 1996) because the linear additive form is a strong assumption in capturing subtleties in each of the component models (see more explanation and analysis in Section 6.2 and Appendix A). The second method is based on **maximum entropy philosophy**, which became very popular in machine learning and natural language processing communities due to the work by Berger, Della Pietra, and Della Pietra (1996), Della Pietra, Della Pietra, and Lafferty (1997), Lau et al. (1993) and Rosenfeld (1996). In fact, for a complete data case, maximum entropy is nothing but maximum likelihood estimation for undirected Markov random fields (MRFs) (Berger, Della Pietra, and Della Pietra 1996; Della Pietra, Della Pietra, and Lafferty 1997). As stated in Wang et al. (2005b), however, there are *two weaknesses* with maximum entropy approach. The first weakness is that this approach can only model distributions over explicitly observed features, but we know there is hidden

information in natural language, such as syntactic structure and semantic topic. The second weakness is that if the statistical model is too complex it becomes intractable to estimate model parameters; computationally very expensive Markov chain Monte Carlo sampling methods (Mark, Miller, and Grenander 1996; Rosenfeld 2000b; Rosenfeld, Chen, and Zhu 2001) would have to be used. One way to overcome the first hurdle is to use a preprocessing tool to extract hidden features (e.g., Rosenfeld [1996] used mutual information clustering method to find word pair triggers) then combine these triggers with trigrams through a maximum conditional entropy approach to allow the discourse topic to influence word prediction; Khudanpur and Wu (2000) used Chelba and Jelinek's structured language model and a word clustering model to extract relevant grammatical and semantic features, then to again combine these features with trigrams through a maximum conditional entropy approach to form a syntactic, semantic, and lexical language model. Wang and colleagues (Wang et al. 2005a; Wang, Schuurmans, and Zhao 2012) have proposed the **latent maximum entropy (LME) principle**, which extends standard maximum entropy estimation by incorporating hidden dependency structure, but still the LME wouldn't overcome the second hurdle. The third method is **directed Markov random field** (Wang et al. 2005b) that overcomes both weaknesses in the maximum entropy approach. Wang et al. used this approach to combine trigram, probabilistic context-free grammar (PCFG), and probabilistic latent semantic analysis (PLSA) models; a generalized inside–outside algorithm is derived that alters the well-known inside–outside algorithm for PCFG (Baker 1979; Lari and Young 1990) with modular modification to take into account the effect of n-gram and PLSA while remaining at the same cubic time complexity. When applying this to the Wall Street Journal corpus with 40 million tokens, they achieved moderate perplexity reduction. Because the probabilistic dependency structure in a structured language model (SLM) (Chelba 2000; Chelba and Jelinek 2000) is more complex and powerful than that in a PCFG, Wang et al. (2006) studied the stochastic properties for the composite language model that integrates n-gram, SLM, and PLSA under the directed MRF framework (Wang et al. 2005b) and derived another *generalized inside–outside* algorithm to train a composite n-gram, SLM, and PLSA language model from a general expectation maximization (EM) (Dempster, Laird, and Rubin 1977) algorithm by following Jelinek's ingenious definition of the inside and outside probabilities for SLM (Jelinek 2004). Again, the generalized inside–outside algorithm alters Jelinek's inside–outside algorithm with modular modification and has the same sixth order of sentence-length time complexity. Unfortunately, there are no experimental results reported.

In this article, we study the same composite n-gram, SLM, and PLSA model under the directed MRF framework as in Wang et al. (2006). The composite n-gram/SLM/PLSA language model under the directed MRF paradigm is first introduced in Section 2. In Section 3, instead of using the sixth order generalized inside–outside algorithm proposed in Wang et al. (2006), we show how to train this composite model via an N-best list approximate EM algorithm that has linear time complexity and a follow-up EM algorithm to improve word prediction power. We prove the convergence of the N-best list approximate EM algorithm. To resolve the data sparseness problem, we generalize Jelinek and Mercer's recursive mixing scheme for Markov source (Jelinek and Mercer 1980) to a mixture of Markov chains. To handle large-scale corpora up to a billion tokens, we demonstrate how to implement these algorithms under a distributed computing environment and how to store this language model on a supercomputer. In Section 4, we describe how to use the model for testing. Related works are then summarized and compared in Section 5. Because language modeling is a data-rich and feature-rich density estimation problem, there is always a trade-off between approximate error

and estimation error, thus in Section 6 we conduct comprehensive experiments on corpora with 44 million tokens, 230 million tokens, and 1.3 billion tokens, and compare perplexity results with n-grams (n = 3, 4, 5 respectively) on these three corpora under various situations; drastic perplexity reductions are obtained. We explain why the composite language models lead to better predictive capacity than linear interpolation. The proposed composite language models are applied to the task of re-ranking the N-best list from Hiero (Chiang 2005, 2007), a state-of-the-art parsing-based machine translation system; we achieve significantly better translation quality measured by the Bleu score and "readability" of translations. Finally, we draw our conclusions and propose future work in Section 7.

The main theme of our approach is "to exploit information, be it syntactic structure or semantic fabric, which involves a fairly high degree of cognition. This is precisely the kind of knowledge that humans naturally and inherently use to process natural language, so it can be reasonably conjectured to represent a key ingredient for success" (Bellegarda 2003, p. 105). In that light, the directed MRF framework, "whose ultimate goal is to integrate all available knowledge sources, appears most likely to harbor a potential breakthrough. It is hoped that the on-going effort conducted in this work to leverage such latent synergies will lead, in the not-too-distant future, to more polyvalent, multi-faceted, effective and tractable solutions for language modeling – this is only beginning to scratch the surface in developing systems capable of deep understanding of natural language" (Bellegarda 2003, p. 105).

2. The Composite n-gram/SLM/PLSA Language Model

Let X denote a set of random variables $(X_\tau)_{\tau\in\Gamma}$ taking values in a (discrete) probability space $(\mathcal{X}_\tau)_{\tau\in\Gamma}$, where Γ is a finite set of states. We define a (discrete) **directed Markov random field** to be a probability distribution $\mathcal{P}$, which admits a recursive factorization if there exist non-negative functions, $\kappa^\tau(\cdot,\cdot), \tau \in \Gamma$ defined on $\mathcal{X}_\tau \times \mathcal{X}_{pa(\tau)}$, such that $\sum_{x_\tau} \kappa^\tau(x_\tau, x_{pa(\tau)}) = 1$ and $\mathcal{P}$ has density

$$p(x) = \prod_{\tau\in\Gamma} \kappa^\tau(x_\tau, x_{pa(\tau)}) \tag{1}$$

Here $pa(\tau)$ denotes the set of parent states of τ. If the recursive factorization refers to a graph, then we have a Bayesian network (Lauritzen 1996). Broadly speaking, however, the recursive factorization can refer to a representation more complicated than a graph with a fixed set of nodes and edges—for example, PCFG and SLM are examples of directed MRFs whose parse tree structure is a random object that can't be described as a Bayesian network (McAllester, Collins, and Pereira 2004). A key difference between directed MRFs and undirected MRFs is that a directed MRF requires many local normalization constraints whereas an undirected MRF has a global normalization factor.

The n-gram (Jelinek 1998; Jurafsky and Martin 2008) language model is essentially a WORD-PREDICTOR, that is, given its entire document history, it predicts the next word $w_{k+1} \in \mathcal{V}$ based on the last n–1 words with probability $p(w_{k+1}|w_{k-n+2}^k)$ where $w_{k-n+2}^k = w_{k-n+2}, \cdots, w_k$ and $\mathcal{V}$ denotes the vocabulary.

The SLM proposed in Chelba and Jelinek (1998, 2000) and Chelba (2000) uses syntactic information beyond the regular n-gram models to capture sentence-level long-range

dependencies. The SLM is based on statistical parsing techniques that allow syntactic analysis of sentences; it assigns a probability $p(W, T)$ to every sentence W and every possible binary parse T. The terminals of T are the words of W with part of speech (POS) tags, and the nodes of T are annotated with phrase headwords and non-terminal labels. Let W be a sentence of length n words to which we have prepended the sentence beginning marker $\langle s \rangle$ and appended the sentence end marker $\langle /s \rangle$ so that $w_0 = \langle s \rangle$ and $w_{n+1} = \langle /s \rangle$. Let $W_k = w_0, \cdots, w_k$ be the word k-prefix of the sentence (the words from the beginning of the sentence up to the current position k) and $W_k T_k$ be the word-parse k-prefix. A word-parse k-prefix has a set of exposed heads $h_{-m}, \cdots, h_{-1} \in \mathcal{H}$, with each head being a pair (headword, non-terminal label), $\mathcal{H} = \mathcal{V} \times \mathcal{O}_{\mathrm{NT}}$ where $\mathcal{O}_{\mathrm{NT}}$ denotes the set of non-terminal label (NTlabel), or in the case of a root-only tree (word, POS tag) $\mathcal{H} = \mathcal{V} \times \mathcal{O}$ where $\mathcal{O}$ denotes the set of POS tags. The exposed heads at a given position k in the input sentence are a function of the word-parse k-prefix.

The SLM operates left-to-right, building up the parse structure in a bottom–up manner. At any given stage of the word generation by the SLM, the exposed headwords are those headwords of the current partial parse which are not yet part of a higher phrase with a head of its own. An mth order SLM (m-SLM) has three operators to generate a sentence:

- The WORD-PREDICTOR predicts the next word $w_{k+1} \in \mathcal{V}$ based on the m most recently exposed headwords $h_{-m}^{-1} = h_{-m}, \cdots, h_{-1}$ in the word-parse k-prefix with probability $p(w_{k+1}|h_{-m}^{-1})$, and then passes control to the TAGGER.
- The TAGGER predicts the POS tag $t_{k+1} \in \mathcal{O}$ to the next word w_{k+1} based on the next word w_{k+1} and the POS tags of the m most recently exposed headwords h_{-m}^{-1} (denoted as $h_{-m}^{-1}.\mathrm{tag} = h_{-m}.\mathrm{tag}, \cdots, h_{-1}.\mathrm{tag}$) in the word-parse k-prefix with probability $p(t_{k+1}|w_{k+1}, h_{-m}^{-1}.\mathrm{tag})$.
- The CONSTRUCTOR builds the partial parse T_{k+1} from T_k, w_{k+1}, and t_{k+1} in a series of moves ending with NULL, where a parse move a is made with probability $p(a|h_{-m}^{-1})$; $a \in \mathcal{A}$={(unary, NTlabel), (adjoin-left, NTlabel), (adjoin-right, NTlabel), NULL}. Depending on an action $a =$ adjoin-right or adjoin-left, the headword h_{-1} or h_{-2} is percolated up by one tree level, the indices of the current exposed headwords $h_{-3}, h_{-4}, \cdots$ are increased by 1, and these headwords together with h_{-1} or h_{-2} become the new exposed headwords. Once the CONSTRUCTOR hits NULL, the headword indexing and current parse structure remain as they are, and the CONSTRUCTOR passes control to the WORD-PREDICTOR.

SLM is thus essentially a generalization of a shift-reduce parser (Aho and Ullman 1972) with *adjoin* corresponding to *reduce* and *predict* to *shift*. (See a detailed description about SLM in Chelba and Jelinek [1998, 2000]; Chelba [2000]; Jelinek [2004]). As an example taken from Jelinek (2004), Figure 1 shows a complete parse where SB/SE is a distinguished POS tag for $\langle s \rangle / \langle /s \rangle$ respectively, ($\langle s \rangle$,TOP) is the only allowed head, and ($\langle /s \rangle$,TOP') is the head of any constituent that dominates $\langle /s \rangle$ but not $\langle s \rangle$. In Figure 1, at the time just after the word *as* is generated, the exposed headwords are "$\langle s \rangle$_SB, show_np, has_vbz." The subsequent model actions are: "POStag as, null, predict its, POStag its, null, predict host, POStag host, adjoin-right-np, adjoin-left-pp, adjoin-left-pp, null, predict a, $\cdots$."

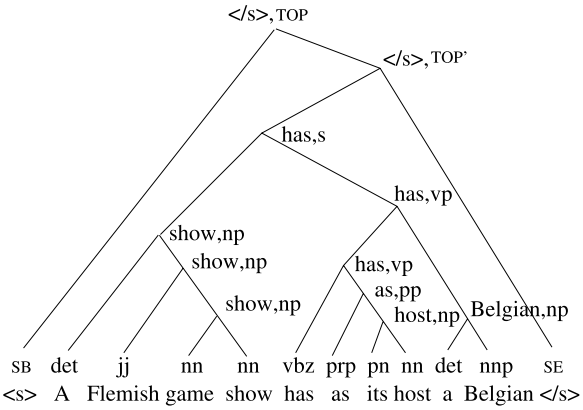

Figure 1
A complete parse tree by the structured language model.

A PLSA model (Hofmann 2001) is a generative probabilistic model of word-document co-occurrences using the bag-of-words assumption described as follows:

- Choose a document d with probability $p(d)$.
- SEMANTIZER selects a semantic class $g \in \mathcal{G}$ with probability $p(g|d)$ where $\mathcal{G}$ denotes the set of topics.
- WORD-PREDICTOR picks a word $w \in \mathcal{V}$ with probability $p(w|g)$.

Because only one pair of (d, w) is being observed, the joint probability model is a mixture of log-linear models with the expression $p(d, w) = p(d) \sum_g p(w|g)p(g|d)$. Typically, the number of documents and the vocabulary size are much larger than the size of latent semantic class variables. Latent semantic class variables therefore function as bottleneck variables to constrain word occurrences in documents.

When combining n-gram, m-SLM, and PLSA together to build a composite generative language model under the directed MRF paradigm (Wang et al. 2005b, 2006), the composite language model is simply a complicated generative model that has four operators: WORD-PREDICTOR, TAGGER, CONSTRUCTOR, and SEMANTIZER. The TAGGER and CONSTRUCTOR in SLM and the SEMANTIZER in PLSA remain unchanged; the WORD-PREDICTORs in n-gram, m-SLM, and PLSA, however, are combined to form a stronger WORD-PREDICTOR that generates the next word, w_{k+1}, not only depending on the m most recently exposed headwords h_{-m}^{-1} in the word-parse k-prefix but also its n-gram history w_{k-n+2}^{k} and its semantic content g_{k+1}. The parameter for WORD-PREDICTOR in the composite n-gram/m-SLM/PLSA language model becomes $p(w|w_{-n+1}^{-1}h_{-m}^{-1}g)$. The resulting composite language model has an even more complex dependency structure but with more expressive power than the original SLM. Figure 2 illustrates the structure of a composite n-gram/m-SLM/PLSA language model.

The composite n-gram/m-SLM/PLSA language model can be formulated as a rather complex chain-tree-table directed MRF model (Wang et al. 2006) with local

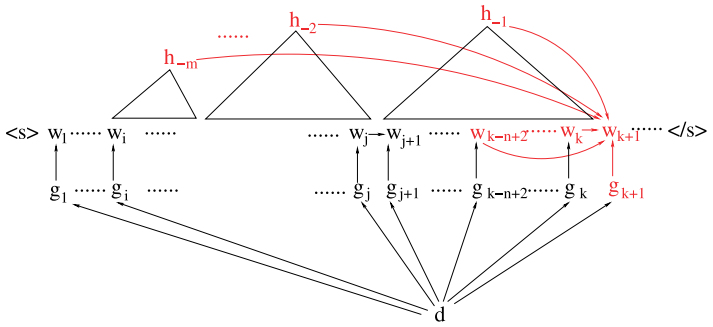

Figure 2
A composite n-gram/m-SLM/PLSA language model where the hidden information is the parse tree T and semantic content g. The n-gram encodes local word interactions, the m-SLM models the sentence's syntactic structure, and the PLSA captures the document's semantic content; all interact together to constrain the generation of natural language. The WORD-PREDICTOR generates the next word w_{k+1} with probability $p(w_{k+1}|w_{k-n+2}^{k}h_{-m}^{-1}g_{k+1})$ instead of $p(w_{k+1}|w_{k-n+2}^{k})$, $p(w_{k+1}|h_{-m}^{-1})$, and $p(w_{k+1}|g_{k+1})$, respectively.

normalization constraints for the parameters of each model component, WORD-PREDICTOR, TAGGER, CONSTRUCTOR, and SEMANTIZER. That is,

$$\sum_{w\in\mathcal{V}} p(w|w_{-n+1}^{-1}h_{-m}^{-1}g) = 1 \tag{2}$$

$$\sum_{t\in\mathcal{O}} p(t|wh_{-m}^{-1}.\text{tag}) = 1 \tag{3}$$

$$\sum_{a\in\mathcal{A}} p(a|h_{-m}^{-1}) = 1 \tag{4}$$

$$\sum_{g\in\mathcal{G}} p(g|d) = 1 \tag{5}$$

If we look at the example in Figure 1, for the composite n-gram/m-SLM/PLSA language model there exists a SEMANTIZER's action to choose a topic g before any WORD-PREDICTOR's action. Moreover, for m-SLM, its WORD-PREDICTOR predicts the next word, such as a, based on m most recently exposed headwords "$\langle s\rangle$-SB, show-np, has-vp," but for the composite model, the WORD-PREDICTOR predicts the next word a based on m most recently exposed headwords "$\langle s\rangle$-SB, show-np, has-vp," n-grams "as its host," and a topic g. These are the only differences between SLM and our proposed composite language model.

3. Training Algorithm

For the composite n-gram/m-SLM/PLSA language model under the directed MRF paradigm, the likelihood of a training corpus $\mathcal{D}$, a collection of documents, can be written as

$$\hat{\mathcal{L}}(\mathcal{D},p) = \prod_{d\in\mathcal{D}} \left(\left(\prod_{l} \left(\sum_{G^l} \left(\sum_{T^l} P_p(W^l,T^l,G^l|d) \right) \right) \right) p(d) \right) \tag{6}$$

where $(W^l, T^l, G^l|d)$ denotes the joint sequence of the lth sentence W^l with its parse structure T^l and semantic annotation string G^l in document d. This sequence is produced by a unique sequence of model actions: WORD-PREDICTOR, TAGGER, CONSTRUCTOR, SEMANTIZER moves; its probability is obtained by chaining the probabilities of these moves

$$P_p(W^l, T^l, G^l|d) = \prod_{g \in \mathcal{G}} \Bigg(p(g|d)^{\#(g, W^l, G^l, d)} \prod_{h_{-1}, \cdots, h_{-m} \in \mathcal{H}} \Bigg(\prod_{w, w_{-1}, \cdots, w_{-n+1} \in \mathcal{V}} p(w|w_{-n+1}^{-1} h_{-m}^{-1} g)^{\#(w_{-n+1}^{-1} w h_{-m}^{-1} g, W^l, T^l, G^l, d)} \prod_{t \in \mathcal{O}} p(t|w h_{-m}^{-1}.\text{tag})^{\#(t, w h_{-m}^{-1}.\text{tag}, W^l, T^l, d)} \prod_{a \in \mathcal{A}} p(a|h_{-m}^{-1})^{\#(a, h_{-m}^{-1}, W^l, T^l, d)} \Bigg) \Bigg) \tag{7}$$

where $\#(g, W^l, G^l, d)$ is the count of semantic content g in semantic annotation string G^l of the lth sentence W^l in document d; $\#(w_{-n+1}^{-1} w h_{-m}^{-1} g, W^l, T^l, G^l, d)$ is the count of n-grams, its m most recently exposed headwords, and semantic content g in parse T^l and semantic annotation string G^l of the lth sentence W^l in document d; $\#(t w h_{-m}^{-1}.\text{tag}, W^l, T^l, d)$ is the count of tag t predicted by word w and the tags of m most recently exposed headwords in parse tree T^l of the lth sentence W^l in document d; and finally $\#(a h_{-m}^{-1}, W^l, T^l, d)$ is the count of constructor move a conditioning on m exposed headwords h_{-m}^{-1} in parse tree T^l of the lth sentence W^l in document d.

Let

$$\mathcal{L}(\mathcal{D}, p) = \prod_{d \in \mathcal{D}} \left(\prod_l \left(\sum_{G^l} \left(\sum_{T^l} P_p(W^l, T^l, G^l|d) \right) \right) \right) \tag{8}$$

then

$$\hat{\mathcal{L}}(\mathcal{D}, p) = \mathcal{L}(\mathcal{D}, p) \prod_{d \in \mathcal{D}} \big(p(d)\big) \tag{9}$$

Clearly, when maximizing $\hat{\mathcal{L}}(\mathcal{D}, p)$ in Equation (6), $p(d)$ is an ancillary term that is independent of all other data-generating parameters, it is not critical to anything that follows; moreover, when a language model is used to find the most likely word sequence in machine translation and speech recognition, this term is useless. Thus, similar to an n-gram language model, we will generally ignore this term and concentrate on optimizing Equation (8) in the subsequent development.

The objective of maximum likelihood estimation is to maximize the likelihood $\mathcal{L}(\mathcal{D}, p)$ with respect to model parameters. For a given sentence, its parse tree and

semantic content are hidden and the number of parse trees grows faster than exponentially with sentence length; Wang et al. (2006) have derived a generalized inside–outside algorithm by applying the standard EM algorithm and considering the auxiliary function

$$Q(p', p) = \sum_{d \in \mathcal{D}} \sum_{l} \sum_{G^l} \sum_{T^l} P_p(T^l, G^l | W^l, d) \log P_{p'}(W^l, T^l, G^l | d) \tag{10}$$

The complexity of this algorithm is sixth order (sentence length), however; thus it is computationally too expensive to be practical for a large corpus even with the use of pruning on charts (Jelinek and Chelba 1999; Jelinek 2004).

3.1 *N*-best List Approximate EM

Similar to SLM (Chelba and Jelinek 1998, 2000; Chelba 2000), we adopt an N-best list approximate EM re-estimation with modular modifications to seamlessly incorporate the effect of n-gram and PLSA components. Instead of maximizing the likelihood $\mathcal{L}(\mathcal{D}, p)$, we maximize the N-best list likelihood,

$$\max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, p, \mathcal{T'}_N) = \prod_{d \in \mathcal{D}} \left(\prod_{l} \left(\max_{\mathcal{T'}^l_N \in \mathcal{T'}_N} \left(\sum_{G^l} \left(\sum_{T^l \in \mathcal{T'}^l_N, ||\mathcal{T'}^l_N|| = N} P_p(W^l, T^l, G^l | d) \right) \right) \right) \right) \tag{11}$$

where $\mathcal{T'}^l_N$ is a set of N parse trees for sentence W^l in document d, $||\cdot||$ denotes the cardinality, and $\mathcal{T'}_N$ is a collection of $\mathcal{T'}^l_N$ for sentences over entire corpus $\mathcal{D}$.

The N-best list approximate EM involves two steps:

1. N-best list search: For each sentence W in document d, find N-best parse trees,

$$\mathcal{T}^l_N = \arg\max_{\mathcal{T'}^l_N} \left\{ \sum_{G^l} \sum_{T^l \in \mathcal{T'}^l_N} P_p(W^l, T^l, G^l | d), ||\mathcal{T'}^l_N|| = N \right\}$$

   and denote $\mathcal{T}_N$ as the collection of N-best list parse trees for sentences over entire corpus $\mathcal{D}$ under model parameter p.

2. EM update: Perform one iteration (or several iterations) of the EM algorithm to estimate model parameters that maximize N-best list likelihood of the training corpus $\mathcal{D}$,

$$\tilde{\mathcal{L}}(\mathcal{D}, p, \mathcal{T}_N) = \prod_{d \in \mathcal{D}} \left(\prod_{l} \left(\sum_{G^l} \left(\sum_{T^l \in \mathcal{T}^l_N \in \mathcal{T}_N} P_p(W^l, T^l, G^l | d) \right) \right) \right)$$

That is,

(a) E-step: Compute the auxiliary function of the N-best list likelihood

$$\tilde{Q}(p', p, \mathcal{T}_N) = \sum_{d \in \mathcal{D}} \sum_{l} \sum_{G^l} \sum_{T^l \in \mathcal{T}_N^l \in \mathcal{T}_N} P_p(T^l, G^l | W^l, d) \log P_{p'}(W^l, T^l, G^l | d)$$

(b) M-step: Maximize $\tilde{Q}(p', p, \mathcal{T}_N)$ with respect to p' to get the new update for p.

Iterate steps (1) and (2) until the convergence of the N-best list likelihood.

We use Zangwill's global convergence theorem (Zangwill 1969) to analyze the behavior of convergence of the N-best list approximate EM.

First, we define two concepts needed for Zangwill's global convergence theorem. A map $\mathcal{M}$ is from points of Θ to subsets of Θ is called a **point-to-set map** on Θ. It is said to be closed at θ if $\theta_i \to \theta, \theta_i \in \Theta$ and $\lambda_i \to \lambda, \lambda_i \in \mathcal{M}(\theta_i)$ implies $\lambda \in \mathcal{M}(\theta)$. For a point-to-point map, continuity implies closedness. Then the global convergence theorem (Zangwill 1969) states the following.

Theorem

Let $\mathcal{M}$ be a point-to-set map (an algorithm) that, given a point $\theta_0 \in \Theta$, generates a sequence $\{\theta_{i=0}^{\infty}\}$ through the iteration $\theta_{i+1} = \mathcal{M}(\theta_i)$. Let $\Omega \in \Theta$ be the set of fixed points of $\mathcal{M}$. Suppose (i) $\mathcal{M}$ is closed over the complement of Ω; (ii) there is a continuous function ϕ on Θ such that (a) if $\theta \notin \Omega$, $\phi(\lambda) > \phi(\theta)$ for all $\lambda \in \mathcal{M}(\theta)$, and (b) if $\theta \in \Omega$, $\phi(\lambda) \geq \phi(\theta)$ for all $\lambda \in \mathcal{M}(\theta)$.

Then all the limit points of $\{\theta_i\}$ are in Ω and $\phi(\theta_i)$ converges monotonically to $\phi(\theta)$ for some $\theta \in \Omega$.

This theorem has been used by Wu (1983) to prove the convergence of a standard EM algorithm (Dempster, Laird, and Rubin 1977). We now use this theorem to show that the N-best list approximate EM algorithm globally converges to the stationary points of the N-best list likelihood. We encounter one difficulty at this point, however, due to the maximization operator in Equation (11); after each iteration the N-best list may have been changed, therefore the set of data presented for the estimation of model parameters may be different from the previous one. Nevertheless, we prove the convergence of the N-best list approximate EM algorithm by checking whether it satisfies two conditions in Zangwill's global convergence theorem. Because the composite model is essentially a mixture model of a curved exponential family through a complex hierarchy, there is a closed form solution for the $\tilde{Q}(p', p, \mathcal{T}_N)$ function irrespective of the N-best list parse trees, so the N-best list approximate EM algorithm is a one-to-one map. Because $\tilde{Q}(p', p, \mathcal{T}_N)$ is continuous in both p' and p, the map is closed, thus condition (i) is satisfied.

To check condition (ii), we need to verify that the N-best list likelihood as a function of p satisfies the properties of $\phi(\theta)$ in condition (ii). Let $\check{\mathcal{T}}_N$ and $\bar{\mathcal{T}}_N$ be the two collections

of N-best list parse trees for sentences over entire corpus $\mathcal{D}$ under two model parameters $\check{p}$ and $\bar{p}$, respectively:

$$\check{\mathcal{T}}_N = \arg\max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, \check{p}, \mathcal{T'}_N) \tag{12}$$

$$\bar{\mathcal{T}}_N = \arg\max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, \bar{p}, \mathcal{T'}_N) \tag{13}$$

and let $\bar{p}$ be the closed form solution of maximizing $\tilde{Q}(p', \check{p}, \check{\mathcal{T}}_N)$ with respect to p', that is,

$$\bar{p} = \arg\max_{p'} \tilde{Q}(p', \check{p}, \check{\mathcal{T}}_N) \tag{14}$$

Then

$$\max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, \bar{p}, \mathcal{T'}_N) \geq \tilde{\mathcal{L}}(\mathcal{D}, \bar{p}, \check{\mathcal{T}}_N) \tag{15}$$

$$\geq \tilde{\mathcal{L}}(\mathcal{D}, \check{p}, \check{\mathcal{T}}_N) \tag{16}$$

$$\geq \max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, \check{p}, \mathcal{T'}_N) \tag{17}$$

The inequality in Equation (15) is strict unless $\check{\mathcal{T}}_N = \bar{\mathcal{T}}_N$, which results in $\bar{p} \in \mathcal{M}(\bar{p})$. Using results proven by Wu (1983), we know that when $\check{p}$ is not a stationary point of the N-best list likelihood or $\check{p} \notin \mathcal{M}(\check{p})$, $\frac{\partial \tilde{\mathcal{L}}(\mathcal{D}, \check{p}, \mathcal{T}_N)}{\partial \check{p}} = \frac{\partial \tilde{Q}(p', \check{p}, \check{\mathcal{T}}_N)}{\partial \check{p}} \neq 0$, $\tilde{Q}(\bar{p}, \check{p}, \check{\mathcal{T}}_N) > \tilde{Q}(\check{p}, \check{p}, \check{\mathcal{T}}_N)$, thus the inequality in Equation (16) is strict. Finally, the inequality in Equation (17) is strict unless $\check{p} \in \mathcal{M}(\check{p})$. Thus condition (ii) is satisfied.

This completes the proof that the N-best list approximate EM algorithm monotonically increases the N-best list likelihood and converges in the sense of Zangwill's global convergence.

In the following, we formally derive the N-best list approximate EM algorithm with linear sentence length time complexity.

3.1.1 N*-best List Search Strategy.* For each sentence W in document d, instead of scanning all the hidden events (both allowed parse trees and semantic annotation strings) we restrict the algorithm to operate with N-best hidden events. We find that, for each document, a large number of topics should be pruned and only a small set of allowed topics should be kept due to the considerations of both computational time and resource demand, otherwise we have to use many more machines to store WORD-PREDICTOR's parameters.

We can either find both the N-best parses for each sentence and N-best topics for each document simultaneously or separately. The latter is much preferred, because the first case is much more computationally expensive.

To extract the N-best topics, we run an EM algorithm for a PLSA model on training corpus $\mathcal{D}$, then keep the N most likely topics (denoted as $\mathcal{G}_d$) according to the values of $p(g|d)$; the rest of the topics are purged.

To extract the N-best parse trees, we adopt a synchronous, multi-stack search strategy that is similar to the one in Chelba and Jelinek (1998, 2000) and Chelba (2000), which involves a set of stacks storing partial parses of the most likely ones for a given prefix W_k and the less probable parses are purged. Each stack contains

hypotheses (partial parses) that have been constructed by the same number of WORD-PREDICTOR and the same number of CONSTRUCTOR operations. The hypotheses in each stack are ranked according to the $\log(P_p(W_k, T_k|d))$ score with the highest on top, where $P_p(W_k, T_k|d) = \sum_{G_k} P_p(W_k, T_k, G_k|d)$ and the W_k, T_k, G_k denote the joint sequence of prefix $W_k = w_0, w_1 \cdots, w_k$ with its parse structure T_k and semantic annotation string $G_k = g_1, \cdots, g_k,\ g_i \in \mathcal{G}_d, i = 1, \cdots, k$ in document d. This sequence is produced by a unique sequence of model actions: WORD-PREDICTOR, TAGGER, CONSTRUCTOR, and SEMANTIZER moves. Its probability is obtained by chaining the probabilities of these moves. The value of $P_p(W_k, T_k|d)$ is computed recursively from $P_p(W_{k-1}, T_{k-1}|d)$ by the following formula:

$$P_p(W_k, T_k|d) = P_p(W_{k-1}, T_{k-1}|d) \left(\sum_{g_k \in \mathcal{G}_d} p(w_k|w_{k-n+1}^{k-1} h_{-m}^{-1} g_k) \frac{p(g_k|d)}{\sum_{g_i \in \mathcal{G}_d} p(g_i|d)} \right) \quad (18)$$
$$p(t_k|w_k, h_{-m}^{-1}.\text{tag}) p(T_{k-1,k}|W_{k-1}T_{k-1}, w_k, t_k)$$

where $W_{k-1}T_{k-1}$ is the word-parse $(k-1)$-prefix; w_k is the kth word predicted by WORD-PREDICTOR; t_k is the tag assigned to w_k by the TAGGER; $T_{k-1,k}$ is the incremental parse structure that generates $T_k = T_{k-1}||T_{k-1,k}$ when attached to T_{k-1}, (this is the parse structure built on top of T_{k-1} and the newly predicted word w_k); the $||$ notation stands for concatenation. Finally, $p(T_{k-1,k}|W_{k-1}T_{k-1}, w_k, t_k)$ is the product of the probabilities of a series of CONSTRUCTOR moves in $T_{k-1,k}$ to form T_k. Because the topics are pruned to $\mathcal{G}_d$, the probability of the SEMANTIZER is normalized to ensure a proper probability distribution. A stack vector consists of the ordered set of stacks containing partial parses with the same number of WORD-PREDICTOR operations but a different number of CONSTRUCTOR operations. In WORD-PREDICTOR and TAGGER operations, some hypotheses are discarded due to the maximum number of hypotheses that the stack can contain at any given time. In the CONSTRUCTOR operation, the resulting hypotheses are discarded due to either finite stack size or the log-probability threshold (the maximum tolerable difference between the log-probability score of the top-most hypothesis and the bottom-most hypothesis at any given state of the stack). The synchronous, multi-stack search strategy is a greedy best-first search algorithm, one of the local heuristic search procedures that does not use future cost estimates to guide the search and thus does not guarantee that the N-best list parse trees are a global optimal solution (Russell and Norvig 2010). In practice, however, we find that the N-best list approximate EM algorithm does converge within several iterations.

3.1.2 EM Update. Once we have both the N-best parse trees for each sentence in document d and the N-best topics for document d, we derive the EM algorithm to estimate model parameters.

Maximizing $\tilde{Q}(p', p, \mathcal{T}_N)$ with respect to p' leads to re-estimated parameters of the composite model, which are nothing but the following normalized conditional expected counts:

$$p'(w|w_{-n+1}^{-1} h_{-m}^{-1} g) \propto \sum_{d \in \mathcal{D}} \sum_{l} \sum_{G^l} \sum_{T^l \in \mathcal{T}_N^l \in \mathcal{T}_N} P_p(T^l, G^l|W^l, d) \#(w_{-n+1}^{-1} w h_{-m}^{-1} g, W^l, T^l, G^l, d) \quad (19)$$

$$p'(t|wh_{-m}^{-1}.\text{tag}) \propto \sum_{d\in\mathcal{D}}\sum_{l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l|W^l,d)\#(twh_{-m}^{-1}.\text{tag},W^l,T^l,d) \tag{20}$$

$$p'(a|h_{-m}^{-1})) \propto \sum_{d\in\mathcal{D}}\sum_{l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l|W^l,d)\#(ah_{-m}^{-1},W^l,T^l,d) \tag{21}$$

$$p'(g|d) \propto \sum_{d\in\mathcal{D}}\sum_{l}\sum_{G^l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l,G^l|W^l,d)\#(g,W^l,G^l,d) \tag{22}$$

In the E-step, we use Equations (19)–(22) to compute the expected count of each model parameter over sentence W^l in document d in the training corpus $\mathcal{D}$. In the full case where the number of parse trees grows faster than exponentially with sentence length, we use Jelinek-style recursive formulas in the generalized inside–outside algorithm (Jelinek 2004) to handle the tree structure and describe the weighted forest of possible derivations (Wang et al. 2006). In the N-best list case considered in this paper, however, we just enumerate each parse tree in the N-best list and compute the expected posterior count for each parse tree. For the WORD-PREDICTOR and the SEMANTIZER, we use Equations (19) and (22) and note that there is a sum over semantic annotation sequence G_l where the number of possible semantic annotation sequences is exponential. We use forward–backward recursive formulas reminiscent of those in hidden Markov models to compute the expected counts. To be more specific, for each parse $T^l \in \mathcal{T}_N^l$, we define the forward vector $\alpha^l(g|d)$ to be

$$\begin{aligned}\alpha_{k+1}^l(g|d) &= \sum_{G_k^l} P_p(W_k^l,T_k^l,w_{k-n+2}^k w_{k+1}h_{-m}^{-1}g,G_k^l|d) \\ &= P_p(W_k^l,T_k^l,w_{k-n+2}^k w_{k+1}h_{-m}^{-1}g|d) \\ &= P_p(W_k^l,T_k^l|d)p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1}g,d)\frac{p(g_{k+1}|d)}{\sum_{g_i\in\mathcal{G}_d}p(g_i|d)}\end{aligned} \tag{23}$$

where W_k^l is the word k-prefix for sentence W^l, and T_k^l is the parse for k-prefix. It is easy to see that the forward vector $\alpha^l(g|d)$ can be recursively computed in a forward manner using Equation (18) as

$$\begin{aligned}\alpha_{k+1}^l(g|d) = &\left(\sum_{g_k\in\mathcal{G}_d}\alpha_k^l(g_k|d)\right)p(t_k|w_k,h_{-m}^{-1}.\text{tag})p(T_{k-1,k}^l|W_{k-1}^l T_{k-1}^l,w_k,t_k) \\ &p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1}g,d)\frac{p(g_{k+1},d)}{\sum_{g_i\in\mathcal{G}_d}p(g_i|d)}\end{aligned} \tag{24}$$

We define the backward vector $\beta^l(g|d)$ to be

$$\beta_{k+1}^l(g|d) = \sum_{G_{k+1,\cdot}^l} P_p(W_{k+1,\cdot}^l,T_{k+1,\cdot}^l,G_{k+1,\cdot}^l|w_{k-n+2}^k w_{k+1}h_{-m}^{-1}g,d) \tag{25}$$

where $W_{k+1,\cdot}^l = w_{k+2}^l,\cdots,\langle/s\rangle$ is the subsequence after word w_{k+1}^l in sentence W^l, $T_{k+1,\cdot}^l$ is the incremental parse structure after the parse structure T_{k+1}^l of word $(k+1)$-prefix

W^l_{k+1} that generates parse tree T^l, $T^l = T^l_{k+1}||T^l_{k+1,\cdot}$, and $G^l_{k+1,\cdot} = g_{k+2}, \cdots$, is the semantic subsequence in G^l relevant to $W^l_{k+1,\cdot}$. Again it is easy to see that the backward vector $\beta^l(g|d)$ can be recursively computed in a backward manner as

$$\begin{aligned}\beta^l_{k+1}(g|d) &= p(t_{k+1}|w_{k+1}, h^{-1}_{-m}.\text{tag})p(T^l_{k,k+1}|W^l_k T^l_k, w_{k+1}, t_{k+1}) \\ &\quad \sum_{g_{k+2}\in\mathcal{G}_d} p(w_{k+2}|w^k_{k-n+3}h^{-1}_{-m}g_{k+2}, d)\frac{p(g_{k+2}|d)}{\sum_{g_i\in\mathcal{G}_d} p(g_i|d)} \\ &\quad \sum_{G^l_{k+2,\cdot}} P_p(W^l_{k+2,\cdot}, T^l_{k+2,\cdot}, G^l_{k+2,\cdot}|w^{k+1}_{k-n+3}w_{k+2}h^{-1}_{-m}g_{k+2}, d) \\ &= p(t_{k+1}|w_{k+1}, h^{-1}_{-m}.\text{tag})p(T^l_{k,k+1}|W^l_k T^l_k, w_{k+1}, t_{k+1}) \\ &\quad \sum_{g_{k+2}\in\mathcal{G}_d} p(w_{k+2}|w^k_{k-n+3}h^{-1}_{-m}g_{k+2}, d)\frac{p(g_{k+2}|d)}{\sum_{g_i\in\mathcal{G}_d} p(g_i|d)}\beta^l_{k+2}(g_{k+2}|d)\end{aligned} \tag{26}$$

Then, the expected count of $w^{-1}_{-n+1}wh^{-1}_{-m}g$ for the WORD-PREDICTOR on sentence W^l in document d is

$$\begin{aligned}&\sum_{G^l}\sum_{T^l\in\mathcal{T}^l_N\in\mathcal{T}_N} P_p(T^l, G^l|W^l, d)\#(w^{-1}_{-n+1}wh^{-1}_{-m}g, W^l, T^l, G^l, d) \\ &= \sum_{G^l}\sum_{T^l\in\mathcal{T}^l_N\in\mathcal{T}_N} P_p(T^l, G^l, W^l|d)\#(w^{-1}_{-n+1}wh^{-1}_{-m}g, W^l, T^l, G^l, d)/P_p(W^l|d) \\ &= \sum_l\sum_k \alpha^l_{k+1}(g|d)\beta^l_{k+1}(g|d)\delta(w^k_{k-n+2}w_{k+1}h^{-1}_{-m}g_{k+1} = w^{-1}_{-n+1}wh^{-1}_{-m}g)/P_p(W^l|d)\end{aligned} \tag{27}$$

where $P_p(W^l|d) = \sum_{G^l}\sum_{T^l\in\mathcal{T}^l_N\in\mathcal{T}_N} P_p(T^l, G^l, W^l|d) = \sum_{T^l\in\mathcal{T}^l_N\in\mathcal{T}_N} P_p(T^l, W^l|d)$, $P_p(T^l, W^l|d)$ is recursively computed by Equation (18) through traversing the lth parse tree $T^l \in \mathcal{T}^l_N$ of sentence W^l from left to right, and $\delta(\cdot)$ is an indicator function. The expected count of g for the SEMANTIZER on sentence W^l in document d is

$$\begin{aligned}&\sum_{G^l}\sum_{T^l\in\mathcal{T}^l_N\in\mathcal{T}_N} P_p(T^l, G^l|W^l, d)\#(g, W^l, G^l, d) \\ &= \sum_l\sum_k \alpha^l_{k+1}(g|d)\beta^l_{k+1}(g|d)p(w_{k+1}|w^k_{k-n+2}h^{-1}_{-m}g)/P_p(W^l|d)\end{aligned} \tag{28}$$

For the TAGGER and the CONSTRUCTOR, we use Equations (20) and (21), and the expected count of each event of twh^{-1}_{-m}.tag and ah^{-1}_{-m} over parse T^l of sentence W^l in document d is the real count appearing in parse tree T^l of sentence W^l in document d times the conditional distribution $P_p(T^l|W^l, d) = P_p(T^l, W^l|d)/\sum_{T^l\in T^l} P_p(T^l, W^l|d)$—that is, $P_p(T^l|W^l, d)\#(twh^{-1}_{-m}.\text{tag}, W^l, T^l, d)$ and $P_p(T^l|W^l, d)\#(ah^{-1}_{-m}, W^l, T^l, d)$, respectively.

When only SLM is considered, the expected count for each model component, WORD-PREDICTOR, TAGGER, and CONSTRUCTOR, over parse T^l of sentence W^l in document d is the real count that appeared in parse T^l of sentence W^l in document d

times the posterior probability $P_p(T^l|W^l, d)$, as is done in Chelba and Jelinek (1998, 2000) and Chelba (2000).

In the M-step, the recursive linear interpolation scheme (Jelinek and Mercer 1980) is used to obtain a smooth probability estimate for each model component (WORD-PREDICTOR, TAGGER, and CONSTRUCTOR). The TAGGER and CONSTRUCTOR are conditional probabilistic models of the type $p(u|z_1, \cdots, z_n)$ where $u, z_1, \cdots, z_n$ belong to a mixed set of words, POS tags, NTtags, and CONSTRUCTOR actions (u only); and $z_1, \cdots, z_n$ form a linear Markov chain. The recursive mixing scheme is the standard one among relative frequency estimates of different orders $k = 0, \cdots, n$ and has been explained in Chelba and Jelinek (1998, 2000) and Chelba (2000). The WORD-PREDICTOR is, however, a conditional probabilistic model $p(w|w_{-n+1}^{-1}h_{-m}^{-1}g)$ where there are three kinds of context, w_{-n+1}^{-1}, h_{-m}^{-1}, and g—each forms a linear Markov chain. The model has a combinatorial number of relative frequency estimates of different orders among three linear Markov chains. We generalize Jelinek and Mercer's (1980) original recursive mixing scheme to handle the situation where the context is a mixture of Markov chains. The factored language (FL) model (Bilmes and Kirchhoff 2003) is close to the smoothing technique we propose here, the major difference is that FL considers all possible combination of the context of conditional probability that can be concisely represented by a factor graph, whereas our approach strictly respects the order of Markov chains for word sequence and headword sequence because we believe natural language tightly follows these orders; moreover, where FL uses a backoff technique, we use linear interpolation.

Consider a composite trigram/2-SLM/PLSA language model. Figure 3 illustrates a lattice formed of all possible conditional probabilistic models and relative frequency

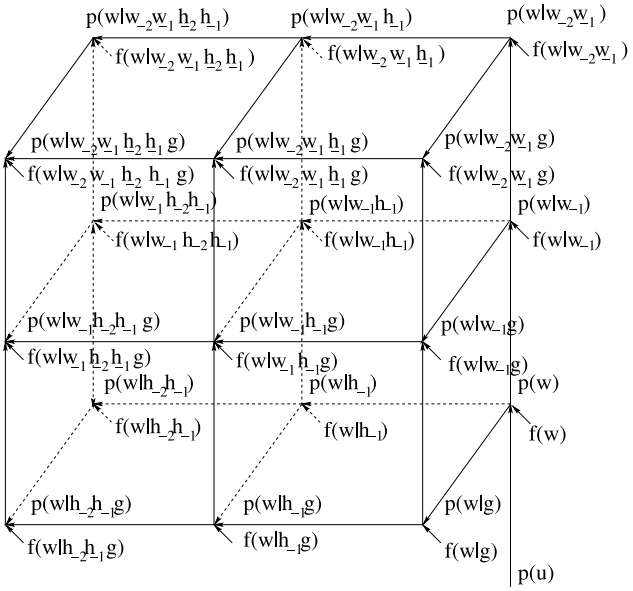

Figure 3
Recursive linear interpolation lattice to estimate WORD-PREDICTOR $p(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$ of the composite trigram/2-SLM/PLSA language model, where $\mathcal{U}$ is the vocabulary in which the predicted random variable w takes values and $p(\mathcal{U})$ denotes uniform distribution of $\mathcal{U}$. The lattice is formed by three linear Markov chains, $w_{-2}w_{-1}$, $h_{-2}h_{-1}$, and g. Starting from $p(\mathcal{U})$, each vertex is visited in a bottom–up, back to front, and right to left order.

estimates of different orders along each of the three linear Markov chains. Each vertex in the lattice represents a conditional probabilistic model that is a linear interpolation of vertices having directed arcs pointing to this vertex and its relative frequency estimate; the linear interpolation coefficients are the weights of directed arcs. For example, the WORD-PREDICTOR $p(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$ is a linear interpolation of three conditional probabilistic models, $p(w|w_{-1}h_{-2}h_{-1}g)$, $p(w|w_{-2}w_{-1}h_{-1}g)$, $p(w|w_{-2}w_{-1}h_{-2}h_{-1})$, and their relative frequency estimate $f(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$,

$$\begin{aligned} p(w|w_{-2}w_{-1}h_{-2}h_{-1}g) = {} & \lambda_w(w_{-2}w_{-1}h_{-2}h_{-1}g) \cdot p(w|w_{-1}h_{-2}h_{-1}g) \\ & +\lambda_h(w_{-2}w_{-1}h_{-2}h_{-1}g) \cdot p(w|w_{-2}w_{-1}h_{-1}g) \\ & +\lambda_g(w_{-2}w_{-1}h_{-2}h_{-1}g) \cdot p(w|w_{-2}w_{-1}h_{-2}h_{-1}) \\ & +(1 - \lambda_w(w_{-2}w_{-1}h_{-2}h_{-1}g) - \lambda_h(w_{-2}w_{-1}h_{-2}h_{-1}g) \\ & -\lambda_g(w_{-2}w_{-1}h_{-2}h_{-1}g)) \cdot f(w|w_{-2}w_{-1}h_{-2}h_{-1}g) \end{aligned} \tag{29}$$

where $\lambda_w(w_{-2}w_{-1}h_{-2}h_{-1}g), \lambda_h(w_{-2}w_{-1}h_{-2}h_{-1}g)$, and $\lambda_g(w_{-2}w_{-1}h_{-2}h_{-1}g)$ are non-negative context-dependent interpolation coefficients with a sum of less than 1; $f(w|w_{-2}w_{-1}h_{-2}h_{-1}g) = \frac{C(w_{-2}w_{-1}wh_{-2}h_{-1}g)}{C(w_{-2}w_{-1}h_{-2}h_{-1}g)}$; and $C(w_{-2}w_{-1}wh_{-2}h_{-1}g)$ is the expected count of the event $w_{-2}w_{-1}wh_{-2}h_{-1}g$ that is extracted from the training corpus by the E-step of the N-best approximate EM algorithm, $C(w_{-2}w_{-1}h_{-2}h_{-1}g) = \sum_{w \in \mathcal{U}} C(w_{-2}w_{-2}wh_{-2}h_{-1}g)$. The linear interpolation coefficients are grouped into equivalence classes (tied) based on the range into which the count falls; the count ranges for each equivalence class, "buckets," are set such that a statistically sufficient number of events fall within that range. In our experiments, we set the count ranges to be the intervals of $2^i, i = 0, 1, \cdots, 10$ (i.e., 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, and ∞). These "tied" interpolation weights are determined by the maximum likelihood estimate from cross-validation data through the EM algorithm (Dempster, Laird, and Rubin 1977) where we use a public available parser in the openNLP software[1] to parse sentences in cross-validation data, and we run LSA to extract N most likely topics for each document in cross-validation data, then we gather joint counts for each model component, WORD-PREDICTOR, TAGGER, CONSTRUCTOR used to determine interpolation weights.

In the M-step, assuming that the count ranges and the corresponding interpolation values for each order are kept fixed to their initial values, the only parameters to be re-estimated using the EM algorithm are the maximal order counts for each model component. The interpolation scheme outlined here is then used to obtain a smooth probability estimate for each model component.

3.2 Follow-up EM

As explained in Chelba and Jelinek (2000) and Chelba (2000), for the SLM component a large fraction of the partial parse trees that can be used for assigning probability to the next word do not survive in the synchronous, multi-stack search strategy, thus they are not used in the N-best approximate EM algorithm for the estimation of WORD-PREDICTOR to improve its predictive power. To remedy this weakness, we estimate a

1 http://www.codeproject.com/KB/recipes/englishparsing.aspx.

separate WORD-PREDICTOR (and SEMANTIZER) model using the partial parse trees exploited by the synchronous, multi-stack search strategy.

First, we look at how to compute the *language model* probability assignment for the word at position k+1 in the input sentence of document d when the word-parse k-prefix W_kT_k is available. From the causal relationship among the parameters of the composite n-gram/m-SLM/PLSA, we have

$$\begin{aligned} P_p(w_{k+1}|W_k,d) &= \sum_{T_k\in Z_k, g_{k+1}\in\mathcal{G}_d} P_p(w_{k+1},T_k,g_{k+1}|W_k,d) \\ &= \sum_{T_k\in Z_k, g_{k+1}\in\mathcal{G}_d} P_p(w_{k+1}|W_k,T_k,g_{k+1},d)P_p(T_k|W_k,d)\frac{p(g_{k+1}|d)}{\sum_{g_i\in\mathcal{G}_d}p(g_i|d)} \\ &= \sum_{h_{-m}^{-1}\in T_k, T_k\in Z_k, g_{k+1}\in\mathcal{G}_d} p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1} g_{k+1})P_p(T_k|W_k,d)\frac{p(g_{k+1}|d)}{\sum_{g_i\in\mathcal{G}_d}p(g_i|d)} \end{aligned} \tag{30}$$

where $P_p(T_k|W_k,d) = \frac{\sum_{G_k} P_p(W_k,T_k,G_k|d)}{\sum_{T_k\in Z_k}\sum_{G_k} P_p(W_k,T_k,G_k|d)}$ to ensure a proper probability normalization over word strings W_k; Z_k is the set of all parses present in the stacks at the current stage k during the synchronous multi-stack pruning strategy and it is a function of the word k-prefix $W_k = w_0,\cdots,w_k$; $G_k = g_1,\cdots,g_k, g_i \in \mathcal{G}_d, i = 1,\cdots,k$ is the semantic string up to k; and $P_p(W_k,T_k,G_k|d)$ is the joint probability of word-parse k-prefix W_kT_k and its semantic string G_k in a document d.

The likelihood of a training corpus $\mathcal{D}$ under this language model probability assignment that uses partial parse trees generated during the process of the synchronous, multi-stack search strategy can be written as

$$\tilde{\mathcal{L}}(\mathcal{D},p) = \prod_{d\in\mathcal{D}}\prod_{l}\Big(P_p(W^l|d)\Big) \tag{31}$$

where $P_p(W^l|d) = \prod_k P_p(w_{k+1}^{(l)}|W_k^l,d)$ and W^l is the lth sentence in document d. Again, similar to Equation (8), we ignore the ancillary term $p(d)$ in Equation (31).

We use a second stage of parameter re-estimation for $p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1} g_{k+1})$ and $p(g_{k+1}|d)$ by maximizing Equation (31) to improve WORD-PREDICTOR's predictive power. In this case, the estimation of the WORD-PREDICTOR is for the emission probability of a hidden Markov model with fixed transition probabilities (although dependent on the position k in the input sentence) specified by the $P_p(T_k|W_k,d)\frac{p(g_{k+1}|d)}{\sum_{g_i\in\mathcal{G}_d}p(g_i|d)}$ values. We use EM again. The E-step is to gather expected joint counts $C(w_{k-n+2}^k w_{k+1} h_{-m}^{-1} g_{k+1}, d)$ and $C(g_{k+1},d)$ of the WORD-PREDICTOR model by accumulating each count at position k weighted by a posterior probability $P_p(T_k,g_{k+1}|w_{k+1},W_k,d)$, namely,

$$P_p(T_k,g_{k+1}|w_{k+1},W_k,d) = \frac{p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1} g_{k+1})p(g_{k+1}|d)P_p(T_k|W_k,d)}{\sum_{h_{-m}^{-1}\in T_k\in Z_k, g\in\mathcal{G}_d} p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1} g)p(g|d)P_p(T_k|W_k,d)}$$

The M-step uses the same count smoothing technique as that described in the N-best list approximate EM.

3.3 Distributed Architecture

When using very large corpora to train our composite language model, the data and the parameters cannot be stored together on a single machine, so we have to resort to distributed computing. The topic of large-scale distributed language models is relatively new, and existing work is restricted to n-grams only (Zhang, Hildebrand, and Vogel 2006; Brants et al. 2007; Emami, Papineni, and Sorensen 2007). Although all existing research use distributed architectures that follow the client–server paradigm, the real implementations are in fact different. Zhang et al. (2006) and Emami et al. (2007) store training corpora in suffix arrays such that one sub-corpus per server serves raw counts, and test sentences are loaded in a client. This implies that when computing the language model probability of a sentence in a client, all servers need to be contacted for each n-gram request. The approach by Brants et al. (2007) follows a standard MapReduce paradigm (Dean and Ghemawat 2004): The corpus is first divided and loaded into a number of clients, and n-gram counts are collected at each client, then the n-gram counts are mapped via hashing and are stored in a number of servers, resulting in exactly one server being contacted per n-gram when computing the language model probability of a sentence. We adopt a similar approach to Brants et al. (2007) and make it suitable to perform iterations of the N-best list approximate EM algorithm (see Figure 4). The corpus is divided and loaded into a number of clients. We use a publicly available parser to parse the sentences in each client to get the initial counts for $w_{-n+1}^{-1}wh_{-m}^{-1}g$ (WORD-PREDICTOR), twh_{-m}^{-1}.tag (TAGGER), and ah_{-m}^{-1} (CONSTRUCTOR), we finish the Map part, and then the counts for a particular $w_{-n+1}^{-1}wh_{-m}^{-1}g$ at different clients are summed up and stored in one of the servers by hashing through word w_{-1}, headword h_{-1}, and its topic g. The counts for all twh_{-m}^{-1}.tag and ah_{-m}^{-1} at different clients are summed up and stored in one of the servers, then we complete the Reduce part. This is the initialization of the N-best list approximate EM step. Each client then calls the servers for parameters to perform a synchronous multi-stack search for each sentence to get the N-best list parse trees. Again, the expected count for a particular parameter of $w_{-n+1}^{-1}wh_{-m}^{-1}g$, twh_{-m}^{-1}.tag, and ah_{-m}^{-1} at the clients are computed, thus we finish the Map part. The expected count of $w_{-n+1}^{-1}wh_{-m}^{-1}g$ are then summed up and stored in one of the servers by hashing through word w_{-1}, headword h_{-1}, and its topic g, and the counts for all twh_{-m}^{-1}.tag and ah_{-m}^{-1} at different clients are summed up and stored in one of the servers; thus we finish the Reduce part. The SEMANTIZER has document-specific

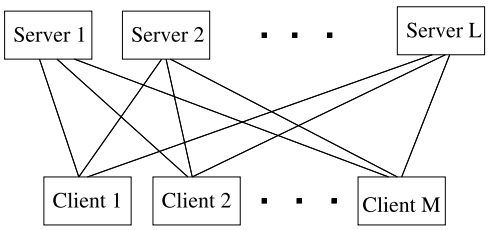

Figure 4
Distributed architecture is essentially a MapReduce paradigm: Clients store partitioned data and perform the E-step: compute expected counts; this is "Map." Servers store parameters (counts) for the M-step where counts of $w_{-n+1}^{-1}wh_{-m}^{-1}g$ are hashed by word w_{-1}, headword h_{-1}, and its topic g to evenly distribute these model parameters into servers as much as possible and counts of twh_{-m}^{-1}.tag and ah_{-m}^{-1} are stored into one server; this is "Reduce."

parameters, thus the EM iterative updates are performed at each of local clients. We repeat this procedure until convergence.

Similarly, we use a distributed architecture as in Figure 4 to perform the follow-up EM algorithm to re-estimate WORD-PREDICTOR.

4. Using the Model for Testing

When a language model is used in one-pass decoders of speech recognition and phrased-based MT systems to guide the search, the search space is organized as a prefix tree and operates left to right, thus we need to know the language model probability at the word level given by Equation (30) one word at a time. Because a document of the test data is not contained in the original training corpus, to compute the language model probability assignment for word w_{k+1} we use a "fold-in" heuristic approach similar to the one used in Hofmann (2001): The parameters corresponding to SEMANTIZER, $p(g|d)$, are re-estimated by maximizing the probability of word subsequence seen so far—that is, a pseudo-document $\tilde{d}_k = (W_k, S)$, where S is the set of previous sentences of a document in test data—while holding the other parameters fixed. Wang et al. (2005b) use on-line gradient ascent to re-estimate these parameters. We use three methods, *one-step on-line EM*, *on-line EM with fixed learning rate*, and *batch EM*, to re-estimate these parameters. Both one-step on-line EM and on-line EM with fixed learning rate use Equation (32) with γ set to $\frac{1}{|\tilde{d}_k|+1}$ and a constant 0.2, respectively.

$$p(g|\tilde{d}_k) = \gamma \frac{\sum_{h_{-m}^{-1} \in T_{k-1}; T_{k-1} \in Z_{k-1}} p(w_k|w_{k-n+1}^{k-1} h_{-m}^{-1} g) p(g|\tilde{d}_{k-1}) P_p(T_{k-1}|W_{k-1}, \tilde{d}_{k-1})}{\sum_{g \in \mathcal{G}_d} \sum_{h_{-m}^{-1} \in T_{k-1}; T_{k-1} \in Z_{k-1}} p(w_k|w_{k-n+1}^{k-1} h_{-m}^{-1} g) p(g|\tilde{d}_{k-1}) P_p(T_{k-1}|W_{k-1}, \tilde{d}_{k-1})}$$
$$+(1-\gamma) p(g|\tilde{d}_{k-1}) \quad (32)$$

The batch EM is the standard EM algorithm where we repeat the iterative procedure until convergence. The initial values are set to $\frac{\sum_{d \in \mathcal{D}} \#(d) \frac{p(g|d)}{\sum_{g_i \in \mathcal{G}_d} p(g_i|d)}}{|\mathcal{D}|}$, where for the topics that are purged we just plug in 0 for $p(g|d)$. $\#(d)$ is the number of words in document $d, d \in \mathcal{D}$, and $|\mathcal{D}| = \sum_d \#(d)$ denotes the size of training corpus (which is the total number of words in the entire training corpus).

When we use Equation (30) to compute perplexity, the system only uses information coming from previous words to generate a topic distribution, which then is used to predict the next word, so the sum over all next words is 1.

We find that the perplexity results are sensitive to these three methods and the initial values. For example, for batch EM, if we set initial values to be those obtained by using the pseudo-document up to the previous word $\tilde{d}_{k-1} = (W_{k-1}, S)$ and trained by batch EM, we obtain worse perplexity results. Table 8 in Section 6.2 gives perplexity results that use these three methods to re-estimate the parameters of the SEMANTIZER, where the on-line EM with fixed learning rate not only has the cheapest computational cost but also leads to the highest perplexity reductions.

5. Related Work

Besides the work by Wang et al. (2005b, 2006) that was discussed in the Introduction, the closest work to ours is that by Khudanpur and Wu (2000) where the authors used

SLM and a word clustering model to extract relevant grammatical and semantic features, then integrated these features with n-grams by a maximum conditional entropy approach. Our composite language model is a generative model, all features play important roles in the EM iterations to allow maximal order events for WORD-PREDICTOR to appear; in Khudanpur and Wu (2000), however, the counts for all events are fixed after feature extraction from SLM and word clustering and no new maximal order events for WORD-PREDICTOR are possibly extracted, this potentially hinders the predictive power of WORD-PREDICTOR. Moreover, the training algorithm in Khudanpur and Wu is computationally expensive. Both methods use the first-stage N-best list approximate EM to extract headwords, thus the complexity is at the same order at this stage; at second stage, however, where we use the follow-up EM, they use the maximum entropy approach. The maximum entropy approach is more expensive, mainly in feature expectation and normalization as well as optimization (such as iterative scaling or the quasi Newton method); ours is quite simple, which is expected relative to frequency estimates with proper smoothing.

The highest reported perplexity reductions are those by Goodman (2001), where the author examines the techniques of caching, clustering, higher-order n-grams, skipping models, and sentence-mixture models in various combinations (mainly linear interpolation). The author *compares to the baseline of a Katz smoothed trigram* with no count cutoffs. On a small training corpus with 100k tokens, a 50% perplexity reduction (1 bit improvement) is obtained. On a larger corpus with 284 million tokens without punctuation, the improvement declines to 38%; we assume that this improvement shrinks to 30% when compared with 4-gram as the baseline.

6. Experimental Results

In this section, we first explain the experimental set-up for our experiments, we then show comprehensive perplexity results in various situations, and we end by reporting the results when we apply the composite language model to the task of re-ranking the N-best list from a state-of-the-art parsing-based machine translation system.

6.1 Experimental Set-up

In previous work (Gildea and Hofmann 1999; Bellegarda 2000; Chelba 2000; Chelba and Jelinek 2000; Charniak 2001; Roark 2001), all complex language models have been trained on relatively small data sets. There is the impression that complex language models only lead to better results than n-grams on small training corpora. For example, Jurafsky and Martin (2008, page 482), state, "We said earlier that statistical parsers can take advantage of longer-distance information than n-grams, which suggests that they might do a better job at language modeling/word prediction. It turns out that if we have a very large amount of training data, a 4-gram or 5-gram is nonetheless still the best way to do language modeling." To verify whether this is true, we have trained our language models using three different training sets: one has 44 million tokens, another has 230 million tokens, and the third has 1.3 billion tokens. An independent test set with 354k tokens is chosen. The independent check data set used to determine the linear interpolation coefficients has 1.7 million tokens for the 44 million token training corpus, and 13.7 million tokens for both the 230 million and 1.3 billion token training corpora. All these data sets are taken from the LDC English Gigaword corpus with nonverbalized punctuation and we remove all punctuation. Table 1 provides the detailed information on how these data sets were chosen from the LDC English Gigaword corpus.

Table 1
The corpora used in our experiments.

	1.3 BILLION TOKEN TRAINING CORPUS
AFP	19940512.0003 ∼ 19961015.0568
AFW	19941111.0001 ∼ 19960414.0652
NYT	19940701.0001 ∼ 19950131.0483
NYT	19950401.0001 ∼ 20040909.0063
XIN	19970901.0001 ∼ 20041125.0119
	230 MILLION TOKEN TRAINING CORPUS
AFP	19940622.0336 ∼ 19961031.0797
APW	19941111.0001 ∼ 19960419.0765
NYT	19940701.0001 ∼ 19941130.0405
	44 MILLION TOKEN TRAINING CORPUS
AFP	19940601.0001 ∼ 19950721.0137
	13.7 MILLION TOKEN CHECK CORPUS
NYT	19950201.0001 ∼ 19950331.0494
	1.7 MILLION TOKEN CHECK CORPUS
AFP	19940512.0003 ∼ 19940531.0197
	354K TOKEN TEST CORPUS
CNA	20041101.0006 ∼ 20041217.0009

These are selected from the LDC English Gigaword corpus. AFP = Agence France-Presse; AFW = Associated Press Worldstream; NYT = New York Times; XIN = Xinhua News Agency; and CNA = Central News Agency of Taiwan denote the sections of the LDC English Gigaword corpus.

The vocabulary sizes in all three cases are:

- word (also WORD-PREDICTOR operation) vocabulary: 60k, open—all words outside the vocabulary are mapped to the ⟨unk⟩ token, these 60k words are chosen from the most frequently occurring words in the 44 million token corpus;
- POS tag (also TAGGER operation) vocabulary: 69, closed;
- non-terminal tag vocabulary: 54, closed;
- CONSTRUCTOR operation vocabulary: 157, closed.

The out-of-vocabulary (OOV) rate on the 44 million, 230 million, 1.3 billion token training corpora is 0.6%, 0.9%, and 1.2%, respectively. The OOV rate on the 1.7 million and 13.7 million token check corpora is 0.6% and 1.3%, respectively. The OOV rate on

Table 2
Statistics about the number of types of n-grams ($n = 3, 4, 5$) on the 44 million, 230 million, and 1.3 billion token corpora.

	n=3	n=4	n=5
44 M	14,302,355	23,833,023	29,068,173
230 M	51,115,539	94,617,433	120,978,281
1.3 B	224,767,319	481,645,099	660,599,586

the 354k token test corpus is 2.0%. Table 2 lists the statistics about the number of types of n-grams on these three corpora.

Similar to SLM (Chelba 2000; Chelba and Jelinek 2000), after the parse undergoes headword percolation and binarization, each model component of WORD-PREDICTOR, TAGGER, and CONSTRUCTOR is initialized from a set of parsed sentences. We use the openNLP software[2] to parse a large number of sentences in the LDC English Gigaword corpus to generate an automatic treebank, which has a slightly different word-tokenization than that of the manual treebank such as the Penn Treebank used in Chelba and Jelinek (2000) and Chelba (2000). For the 44 and 230 million token corpora, all sentences are automatically parsed and used to initialize model parameters, whereas for the 1.3 billion token corpus, we parse the sentences from a portion of the corpus that contains 230 million tokens, then use them to initialize model parameters. The parser at openNLP is trained on the Penn Treebank, which has only one million tokens, and there is a mismatch between the Penn Treebank and the LDC English Gigaword corpus. Nevertheless, experimental results show that this approach is effective to provide initial values of model parameters.

6.2 Perplexity Results

Table 3 gives the perplexity results (Bahl et al. 1977) of n-grams (n = 3, 4, and 5) using linear interpolation and Kneser-Ney (1995) smoothing when the training corpus has 44 million, 230 million, and 1.3 billion tokens, respectively. We have implemented a distributed n-gram with linear interpolation smoothing, but we don't have distributed n-grams with Kneser-Ney smoothing implemented by us. Instead, we use the SRI Language Modeling Toolkit to obtain perplexity results of n-grams with Kneser-Ney smoothing for the 44 million and 230 million token corpora using a single machine that has 20G memory at the Ohio Supercomputer center. We are not able to compute perplexity results of n-grams with Kneser-Ney smoothing on the 1.3 billion token corpus, thus we leave these results blank in Table 3. From the results in Table 3, we decided to use a linearly smoothed trigram as the baseline model for the 44 million token corpus, a linearly smoothed 4-gram as the baseline model for the 230 million token corpus, and a linearly smoothed 5-gram as the baseline model for the 1.3 billion token corpus.

As we mentioned in Section 3.1.1, we can keep only a small set of topics due to the considerations of computational time and resource demand. Table 4 shows the perplexity results and computation time of composite n-gram/PLSA language models that are trained on the three corpora when the pre-defined number of total topics is 200, but different numbers of most-likely topics are kept for each document in PLSA; the

2 http://www.codeproject.com/KB/recipes/englishparsing.aspx.

Table 3
Perplexity results of n-grams (n = 3, 4, and 5) using linear interpolation and Kneser-Ney smoothing when training set is a 44 million, 230 million, or 1.3 billion token corpus, respectively.

44 M	LINEAR	KNESER-NEY
n=3	262	244
n=4	258	235
n=5	260	235
230 M	**LINEAR**	**KNESER-NEY**
n=3	217	195
n=4	200	183
n=5	201	183
1.3 B	**LINEAR**	**KNESER-NEY**
n=3	161	—
n=4	141	—
n=5	138	—

Table 4
Perplexity (ppl) results and time consumed of the composite n-gram/PLSA language model trained on three corpora when different numbers of most-likely topics are kept for each document in PLSA.

CORPUS	n	# OF TOPICS	PPL	TIME (HOURS)	# OF SERVERS	# OF CLIENTS	# OF TYPES OF $ww_{-n+1}^{-1}g$
44M	3	5	196	0.5	40	100	120.1M
	3	10	194	1.0	40	100	218.6M
	3	20	190	2.7	80	100	537.8M
	3	50	189	6.3	80	100	1.123B
	3	100	189	11.2	80	100	1.616B
	3	200	188	19.3	80	100	2.280B
230M	4	5	146	25.6	280	100	0.681B
1.3B	5	2	111	26.5	400	100	1.790B
	5	5	102	75.0	400	100	4.391B

rest are pruned. For the composite 5-gram/PLSA model trained on the 1.3 billion token corpus, 400 cores have to be used to keep the top five most likely topics. For the composite trigram/PLSA model trained on the 44M token corpus, the computation time increases drastically, with less than 5% percent perplexity improvement. In the following experiments, therefore, we keep the top five topics for each document from a total of 200 topics—all other 195 topics are pruned.

All composite language models are first trained by performing the N-best list approximate EM algorithm until convergence, then the EM algorithm for a second stage of parameter re-estimation for WORD-PREDICTOR and SEMANTIZER until convergence. We fix the size of topics in the PLSA to be 200 and then prune to 5 in the experiments, where the unpruned 5 topics in general account for 70% probability in $p(g|d)$. Table 5 shows comprehensive perplexity results for a variety of different

Table 5
Perplexity results for various language models on test corpora, where + denotes linear combination, / denotes composite model; n denotes the order of the n-gram, and m denotes the order of the SLM; the topic nodes are pruned from 200 to 5.

LANGUAGE MODEL	44M n=3,m=2	REDUCTION	230M n=4,m=3	REDUCTION	1.3B n=5,m=4	REDUCTION
BASELINE n-GRAM (LINEAR)	262		200		138	
n-GRAM (KNESER-NEY)	244	6.9%	183	8.5%	—	—
m-SLM	279	−6.5%	190	5.0%	137	0.0%
PLSA	825	−214.9%	812	−306.0%	773	−460.0%
n-GRAM+m-SLM	247	5.7%	184	8.0%	129	6.5%
n-GRAM+PLSA	235	10.3%	179	10.5%	128	7.2%
n-GRAM+m-SLM+PLSA	222	15.3%	175	12.5%	123	10.9%
n-GRAM/m-SLM	243	7.3%	171	14.5%	(125)	9.4%
n-GRAM/PLSA	196	25.2%	146	27.0%	102	26.1%
m-SLM/PLSA	198	24.4%	140	30.0%	(103)	25.4%
n-GRAM/PLSA+m-SLM/PLSA	183	30.2%	140	30.0%	(93)	32.6%
n-GRAM/m-SLM+m-SLM/PLSA	183	30.2%	139	30.5%	(94)	31.9%
n-GRAM/m-SLM+n-GRAM/PLSA	184	29.8%	137	31.5%	(91)	34.1%
n-GRAM/m-SLM+n-GRAM/PLSA+m-SLM/PLSA	180	31.3%	130	35.0%	—	—
n-GRAM/m-SLM/PLSA	176	32.8%	—	—	—	—

models such as composite n-gram/m-SLM, n-gram/PLSA, m-SLM/PLSA, their linear combinations, and so on, where we use on-line EM with a fixed learning rate to re-estimate the parameters of the SEMANTIZER of test document. The m-SLM performs competitively with its counterpart n-gram ($n = m+1$) on large scale corpus. Table 6 lists the statistics about the number of types in the predictor of the m-SLMs on these three corpora, where for the 230 million token and 1.3 billion token corpora we cut off the fractional expected counts that are less than a predefined threshold of 0.005, to significantly reduce the number of the predictor's types by 70%.

In Table 5, for the composite n-gram/m-SLM model ($n = 3, m = 2$ and $n = 4, m = 3$) trained on 44 million tokens and 230 million tokens, we cut off its fractional expected counts that are less than a threshold 0.005; this significantly reduces the number of the predictor's types by 85%. When we train the composite language on the 1.3 billion token corpus, we have to both aggressively prune the parameters of WORD-PREDICTOR and shrink the order of n-gram and m-SLM in order to store them in a supercomputer having 1,000 cores. In particular, for the composite 5-gram/4-SLM model, its size is too big to store, thus we use its approximation, a linear combination of 5-gram/2-SLM and 2-gram/4-SLM. For the 5-gram/2-SLM or 2-gram/4-SLM, again we cut off its fractional expected counts that are less than a threshold 0.005, which significantly reduces the number of the predictor's types by 85%. For the composite 4-SLM/PLSA model, we cut off its fractional expected counts that are less than a threshold 0.002, again this significantly reduces the number of predictor's types by 85%. For the composite 4-SLM/PLSA model or its linear combination with models, we ignore all the tags and use only the words in the four headwords. We have checked that the conditional language model (Equation [30]) sums to 1 for large randomly selected conditional events. The composite n-gram/m-SLM/PLSA model gives significant perplexity reductions over baseline n-grams ($n = 3, 4, 5$) and m-SLMs ($m = 2, 3, 4$). The majority of gains comes from the PLSA component, but when adding the SLM component into the n-gram/PLSA, there is a further 10% relative perplexity reduction.

Table 7 shows how large the composite 5-gram/PLSA, 5-gram/2-SLM (or 2-gram/4-SLM), and 4-SLM/PLSA models are when trained by the 1.3 billion token corpus after aggressive pruning. The total minimum number of servers used to store the parameters of the predictor for the composite 5-gram/PLSA, 5-gram/2-SLM (or 2-gram/4-SLM), and 4-SLM/PLSA models is, respectively, 400, 240, 400, and the number of clients to store the partitioned data of the 1.3 billion token corpus is 100 for these three composite language models. There is no way to store the parameters of the linear combination of the composite 5-gram/PLSA, 5-gram/2-SLM (or 2-gram/4-SLM), and 4-SLM/PLSA models in our currently available supercomputer resources.

Table 6
Statistics about the number of types in the predictor of the m-SLMs ($m = 2, 3, 4$) on the 44 million, 230 million, and 1.3 billion token corpora. For the 230 million and 1.3 billion token corpora, fractional expected counts that are less than a threshold are pruned to significantly reduce the number of m-SLM (m=3, 4) predictor's types by 70%.

	m=2	m=3	m=4
44 M	189,002,525	269,685,833	318,174,025
230 M	267,507,672	1,154,020,346	1,417,977,184
1.3 B	946,683,807	1,342,323,444	1,849,882,215

Table 7
Counts of the types in the predictor of the 5-gram/PLSA, 5-gram/2-SLM (or 2-gram/4-SLM), and 4-SLM/PLSA models when trained on the 1.3B corpus. Fractional expected counts that are less than a threshold are pruned; this significantly reduces the number of predictor's types by 85%.

COMPOSITE MODEL	TYPES OF	# OF TYPES	# OF SERVERS	# OF CLIENTS
5-GRAM/PLSA	$w_{-4}^{-1}wg$	4.39 B	400	100
5-GRAM/2-SLM	$w_{-4}^{-1}wh_{-2}^{-1}$	2.01B	240	100
2-GRAM/4-SLM	$w_{-1}wh_{-4}^{-1}$			
4-SLM/PLSA	$wh_{-4}^{-1}g$	4.88 B	400	100

Appendix A shows an example of sentence probability that is provided by 5-gram, 5-gram/PLSA, and 5-gram/4-SLM+5-gram/PLSA models, respectively; these language models are trained using the 1.3 billion tokens corpus. The example demonstrates that our composite model is able to extract topic information and grammatical structure to improve word prediction for natural language.

Table 8 shows the perplexity results for composite n-gram/PLSA and n-gram/m-SLM/PLSA language models when three methods are used to re-estimate the parameters of the SEMANTIZER of test document; we use superscript 1, 2, and 3 to denote that during testing we used one step on-line EM, on-line EM with fixed learning rate, and batch EM, respectively. The on-line EM with fixed learning rate gives the best perplexity results as well as the least computation time. Again, when we train the composite language on the 1.3 billion token corpus, we have to shrink the order of the n-gram and m-SLM in order to store them in a supercomputer having 1,000 cores. For the composite 4-SLM/PLSA model or its linear combination with models, we ignore all the tags and use only the words in the four headwords. For the composite 5-gram/4-SLM model or its linear combination with models, we in fact use its approximation, a linear combination of the 5-gram/2-SLM and 2-gram/4-SLM models.

To better explain and analyze our model, we mark the perplexity results for the 40 million token corpus in Table 5 on the vertices in Figure 3 to reveal many insights. The baseline trigram result is given by the vertex $p(w|w_{-2}w_{-1})$, the 2-SLM result is given by the vertex $p(w|h_{-2}h_{-1})$, the PLSA result is given by the vertex $p(w|g)$, the trigram/2-SLM result is given by the vertex $p(w|w_{-2}w_{-1}h_{-2}h_{-1})$, the trigram/PLSA result is given by the vertex $p(w|w_{-2}w_{-1}g)$, and the trigram/2-SLM/PLSA is given by the vertex $p(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$. The trigram+2-SLM result is given by a linear combination of vertices $p(w|w_{-2}w_{-1})$ and $p(w|h_{-2}h_{-1})$; the trigram+PLSA result is given by a linear combination of vertices $p(w|w_{-2}w_{-1})$ and $p(w|g)$; and the trigram+2-SLM+PLSA result is given by a linear combination of vertices $p(w|w_{-2}w_{-1})$, $p(w|h_{-2}h_{-1})$, and $p(w|g)$. The trigram/PLSA+2-SLM/PLSA result is given by a linear combination of vertices $p(w|w_{-2}w_{-1}g)$ and $p(w|h_{-2}h_{-1}g)$, and so on. The trigram/PLSA+trigram/2-SLM+2-SLM/PLSA result is given by a linear combination of vertices $p(w|w_{-2}w_{-1}g)$, $p(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$, and $p(w|h_{-2}h_{-1}g)$. The composite trigram/2-SLM/PLSA language model is more powerful and expressive than the linear combination of trigram, 2-SLM, and PLSA for two reasons. First, valuable relative frequency estimates such as $f(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$,$f(w|w_{-2}w_{-1}h_{-2}h_{-1})$, and so forth, are encoded into the composite language model, as seen from Figure 3. As long as there are events such as $w_{-2}w_{-1}wh_{-2}h_{-1}g$, and so on, that occur explicitly or implicitly in the training

Table 8
Perplexity results for the composite n-gram/PLSA and n-gram/m-SLM/PLSA language models on the test corpus, where + denotes linear combination, / denotes composite model; n is the order of the n-gram and m is the order of the SLM, and superscripts 1, 2, 3 denote using one-step on-line EM, on-line EM with fixed learning rate, and batch EM during testing, respectively.

LANGUAGE MODEL	44M n=3,m=2	REDUCTION	230M n=4,m=3	REDUCTION	1.3B n=5,m=4	REDUCTION
n-GRAM (LINEAR)	262		200		138	
n-GRAM/PLSA^1	202	22.9%	150	25.0%	107	22.5%
n-GRAM/m-SLM+n-GRAM/PLSA^1	192	26.7%	142	29.0%	(97)	29.1%
n-GRAM/PLSA^2	196	25.2%	146	27.0%	102	26.1%
n-GRAM/m-SLM+n-GRAM/PLSA^2	184	29.8%	137	31.5%	(91)	34.1%
n-GRAM/PLSA^3	201	23.3%	148	26.0%	104	24.6%
n-GRAM/m-SLM+n-GRAM/PLSA^3	189	27.9%	140	30.0%	(92)	33.3%

corpus, the composite trigram/2-SLM/PLSA will take them into account to improve the prediction power for test data, whereas a linear combination of trigram, 2-SLM, and PLSA just neglects a large amount of this valuable information. The second reason is that the weights used in a simple linear combination are context-independent, thus more restricted. Similarly, the composite trigram/2-SLM/PLSA language model is more powerful and expressive than a linear combination of pairwise composite language models (e.g., trigram/2-SLM, trigram/PLSA, and 2-SLM/PLSA), since the composite trigram/2-SLM/PLSA can take advantage of the relative frequency estimate $f(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$,$f(w|w_{-2}w_{-1}h_{-1}g)$, and $f(w|w_{-1}h_{-2}h_{-1}g)$. The improvement in this case shrinks, however, because pairwise composite language models use some valuable lower order relative frequency estimates such as $f(w|w_{-2}w_{-1}g)$, and so forth. Stated another way, each vertex of the lattice in Figure 3 is an expert of WORD-PREDICTOR that is proficient in making a prediction based on the context represented at the vertex; it predicts words based on the information provided by a committee consisting of experts from parent vertices as well as the relative frequency estimate it extracts. These experts are hierarchically organized, with the WORD-PREDICTOR of the composite trigram/2-SLM/PLSA (i.e., $p(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$) overseeing all available information to make the most powerful prediction.

Finally, we conducted experiments where we fixed the size of the training data and increased the complexity of our language models. Because available resources are limited, preventing us from considering complex language models that are trained on the 1.3 billion token corpus, we considered complex language models trained on the 44 million token corpus instead. Table 9 shows the perplexity results. We can see that as we increase the order for n-gram and m-SLM from $n = 3$ and $m = 2$ to $n = 4$ and $m = 3$, the composite language models become better and have up to 5% perplexity reductions; when we increase the order for n-gram and m-SLM to $n = 5$ and $m = 4$, however, the composite language models become worse and slightly overfit the data even if we use linear interpolation smoothing, and there are no further perplexity reductions.

To summarize, as a sub-problem for MT and speech recognition under the source-channel paradigm (Jelinek 2009), language modeling is a data-rich and feature-rich density estimation problem with Kullback-Leibler divergence as a cost function, and there is always a trade-off between approximation error and estimation error (Barron and Sheu 1991), reminiscent of the "bias-variance" trade-off for a regression problem with a quadratic cost function (Hastie, Tibshirani, and Friedman 2009). Figure 5 explains the perplexity results in Tables 3 and 5 from a model selection point of view.

Let $\hat{p}$ denote the true (but unknown) distribution of natural language, its information projection to n-grams is the minimum Kullback-Leibler divergence from $\hat{p}$ to n-grams (Amari and Nagaoka 2000; Wang, Greiner, and Wang 2009) and is denoted as $\hat{p}_n, n = 3, 4, 5, 6$. Let $\tilde{p}$ denote the empirical distribution of natural language—in particular, $\tilde{p}_M$ denotes the empirical distribution for a million token corpus, $\tilde{p}_B$ denotes the empirical distribution for a billion token corpus, and $\tilde{p}_T$ denotes the empirical distribution for a trillion token corpus. The information projection of $\tilde{p}_M$ to trigram is p^3_M, to 4-gram is p^4_M, and to 5-gram is p^5_M. The distance between $\hat{p}$ and $\hat{p}_n (n = 3, 4, 5, 6)$, $D(\hat{p}, \hat{p}_n)$, is the approximation error when using n-gram to represent $\hat{p}$, that is, the best the n-gram can do when abundant data are available. The distance between $\tilde{p}^n_M$ and $\hat{p}_n, n = 3, 4, 5$, $D(\tilde{p}^n_M, \hat{p}_n)$, is the estimation error when only the million token corpus is available. The Pythagorean theorem states that the distance between $\hat{p}$ and $\tilde{p}_M$, $D(\hat{p}, \tilde{p}_M)$, is the sum of the approximation error and the estimation error (Barron and Sheu 1991; Amari and Nagaoka 2000; Wang, Greiner, and Wang 2009). In language modeling research, because $\hat{p}$ is unknown, the distance between $\hat{p}$ and $p^n_M, n = 3, 4$ is approximately computed

Table 9
Perplexity results for various language models on test corpora, where + denotes linear combination, / denotes composite model; n denotes the order of the n-gram, and m denotes the order of the SLM; the topic nodes are pruned from 200 to 5.

LANGUAGE MODEL	44M n=3,m=2	REDUCTION	44M n=4,m=3	REDUCTION	44M n=5,m=4	REDUCTION
BASELINE n-GRAM (LINEAR)	262		258		260	
n-GRAM (KNESER-NEY)	244	6.9%	235	8.9%	235	9.6%
m-SLM	279	−6.5%	254	1.6%	254	2.3%
n-GRAM+m-SLM	247	5.7%	233	9.7%	234	10.0%
n-GRAM+PLSA	235	10.3%	230	10.9%	231	11.2%
n-GRAM+m-SLM+PLSA	222	15.3%	220	14.7%	221	15.0%
n-GRAM/m-SLM	243	7.3%	232	10.1%	235	9.6%
n-GRAM/PLSA	196	25.2%	189	26.7%	193	25.8%
m-SLM/PLSA	198	24.4%	190	26.4%	192	26.2%
n-GRAM/PLSA+m-SLM/PLSA	183	30.2%	179	30.6%	178	31.5%
n-GRAM/m-SLM+m-SLM/PLSA	183	30.2%	178	31.0%	180	30.8%
n-GRAM/m-SLM+n-GRAM/PLSA	184	29.8%	176	31.8%	178	31.5%
n-GRAM/m-SLM+n-GRAM/PLSA+m-SLM/PLSA	180	31.3%	173	33.0%	173	33.5%
n-GRAM/m-SLM/PLSA	176	32.8%	169	34.5%	171	34.2%

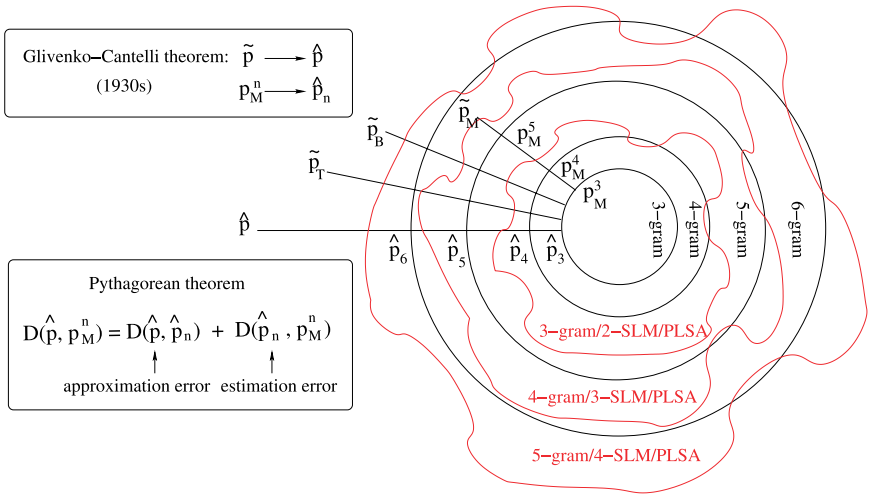

Figure 5
Language modeling is a data-rich and feature-rich density estimation problem. The information projection from true distribution and empirical distribution to n-grams is unique, and the information projection from true distribution and empirical distribution to composite language models might be local optimal. There is a trade-off between approximation error and estimation error for composite language models.

by the perplexity result using test data. By the Glivenko-Cantelli theorem (Vapnik 1998), we know that the empirical distribution $\tilde{p}$ converges to the true distribution $\hat{p}$; similarly, the information projection of empirical distribution on an n-gram converges to the information projection on an n-gram of true distribution (i.e., the estimation error shrinks to 0). In the same vein, we can define the information projection of $\hat{p}$ or $\tilde{p}$ to the composite language models and the corresponding approximate error and estimation error, and so forth. In this case, the Pythagorean theorem breaks down due to the non-convexity of the set of composite language models. As noted by Dr. Ciprian Chelba in our private communication on March 20th, 2010, "When playing with large data, the model capacity is an important factor to language model performance: The supply of more data needs to be matched by demand on the model side. A simple way to achieve this in n-grams is to increase the order n as much as the data will allow. This of course implies that the computational aspects of storing and serving such models are solved and that it is not a constraint" (see also Chelba et al. 2010). This is also true for our composite language models as justified from the results in Tables 5 and 9: The composite n-gram/m-SLM/PLSA language model has rich features, thus has smaller approximation error than the n-gram, m-SLM, PLSA, or any composite model of two, or their linear combinations. Table 5 shows that the information projection of the empirical distribution for the million and billion token corpora, $\tilde{p}_M$ and $\tilde{p}_B$ on the composite n-gram/m-SLM/PLSA language model, is closer to the true distribution $\hat{p}$. This is reflected approximately by the perplexity results on test data.

6.3 Re-ranking Machine Translation Results

We have applied our composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA[1] language model that is trained by a 1.3 billion word corpus for the task of re-ranking the N-best list in statistical MT. We used the same two 1,000-best lists that were used

Table 10
10-fold cross-validation Bleu score results for the task of re-ranking the 1,000-best list generated on 919 sentences of 100 documents from the MT03 Chinese–English evaluation set.

SYSTEM MODEL	MEAN (%)	95% CI (%)
BASELINE	31.75	0.22
5-GRAM	32.53	0.24
5-GRAM/2-SLM+2-GRAM/4-SLM	32.87	0.24
5-GRAM/PLSA[1]	33.01	0.24
5-GRAM/2-SLM+2-GRAM/4-SLM+5-GRAM/PLSA[1]	33.32	0.25

by Zhang and colleagues (Zhang, Hildebrand, and Vogel 2006; Zhang 2008; Zhang et al. 2011). The first list was generated on 919 sentences of 100 documents from the MT03 Chinese–English evaluation set, and the second was generated on 191 sentences of 20 documents from the MT04 Chinese–English evaluation set, both by Hiero (Chiang 2007), a state-of-the-art parsing-based translation model. Its decoder uses a trigram language model trained with modified Kneser-Ney smoothing (Jurafsky and Martin 2008) on a 200 million token corpus. Each translation has 11 features and language model is one of them. We substitute our language model and use MERT (Och 2003) to optimize the Bleu score (Papineni et al. 2002). We conduct two experiments on these two data sets. In the first experiment, we partition the first data set that consists of 100 documents into ten pieces; each piece consists of 10 documents, nine pieces are used as training data to optimize the Bleu score (Papineni et al. 2002) by MERT (Och 2003), and the remaining single piece is used to re-rank the 1,000-best list and obtain the Bleu score. The cross-validation process is then repeated 10 times (the folds), with each of the 10 pieces used exactly once as the validation data. The 10 results from the folds then can be averaged (or otherwise combined) to produce a single estimation for Bleu score. The mean and variance of the Bleu score are calculated with each different LM. We assume that the score follows Student's t-distribution and we compute the 95% confidence interval according to mean and variance. Table 10 shows the Bleu scores through 10-fold cross-validation. The composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA[1] language model gives 1.57 percentage point Bleu score improvement over the baseline and 0.79 percentage point Bleu score improvement over the 5-gram. We are not able to further improve Bleu score when we use either the 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA[2] or 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA[3]. This is because there is not much diversity on the 1,000-best list, and essentially only $20 \sim 30$ distinct sentences are in the 1,000-best list.

In the second experiment, we used the first data set as training data to optimize the Bleu score by MERT, then the second data set is used to re-rank the 1,000-best list and obtain the Bleu score. To obtain the confidence interval of the Bleu score, we resort to the bootstrap resampling described by Koehn (2004). We randomly select 10 re-ranked documents from the 20 re-ranked documents in the second data set with replacement. We draw the translation results of the 10 documents and compute the Bleu score. We repeat this procedure 1,000 times. When we compute the 95% confidence interval, we drop the top 25 and bottom 25 Bleu scores, and only consider the range of 26th to 975th Bleu scores. Table 11 shows the Bleu scores. These statistics are computed with different language models, but on the same chosen test sets. The 5-gram gives 0.51 percentage point Bleu score improvement over the baseline. The composite 5-gram/2-SLM+

Table 11
Bleu score results for the task of re-ranking the 1,000-best list generated on 191 sentences of 20 documents from the MT04 Chinese–English evaluation set.

SYSTEM MODEL	MEAN (%)	95% CI (%)
BASELINE	27.59	0.31
5-GRAM	28.10	0.32
5-GRAM/2-SLM+2-GRAM/4-SLM	28.34	0.32
5-GRAM/PLSA[1]	28.53	0.31
5-GRAM/2-SLM+2-GRAM/4-SLM+5-GRAM/PLSA[1]	28.78	0.31

2-gram/4-SLM+5-gram/PLSA[1] language model gives 1.19 percentage point Bleu score improvement over the baseline and 0.68 percentage point Bleu score improvement over the 5-gram.

Chiang (2007) studied the performance of machine translation on Hiero, the Bleu score is 33.31% when *n*-gram is used to re-rank the *N*-best list; the Bleu score becomes significantly higher (37.09%) when the *n*-gram is embedded directly into Hiero's one pass decoder, however. This is because there is not much diversity in the *N*-best list. It is expected that putting our composite language into a one-pass decoder should result in much improved Bleu scores.

Besides reporting the Bleu scores, we look at the "readability" of translations, similar to the study conducted by Charniak, Knight, and Yamada (2003). The translations are sorted into four groups: good/bad syntax crossed with good/bad meaning by human judges (see Table 12). We find that many more sentences are perfect, many more are grammatically correct, and many more are semantically correct. The syntactic language model (Charniak et al. 2003) only improves translations to have good grammar, but does not improve translations to preserve meaning. The composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA[1] language model improves both significantly. Bear in mind that Charniak et al. (2003) integrated Charniak's language model with the syntax-based translation model proposed by Yamada and Knight (2001) to rescore a tree-to-string translation forest, whereas we use only our language model for *N*-best list re-ranking. Also, the same study (Charniak et al. 2003) found that the outputs produced using the *n*-grams received higher scores from Bleu; ours did not. The difference between human judgments and Bleu scores indicates that closer agreement may be possible by incorporating syntactic structure and semantic information into the Bleu score evaluation. For example, semantically similar words like *insure* and *ensure* as in Bleu paper (Papineni et al. 2002) should be substituted in the formula, and there is a weight to measure the goodness of syntactic structure. This modification will lead to a better metric and such information can be provided by our composite language models.

Table 12
Results of "readability" evaluation on 919 translated sentences of 100 documents. P = perfect; S = only semantically correct; G = only grammatically correct; W = wrong.

SYSTEM MODEL	P	S	G	W
BASELINE	95	398	20	406
5-GRAM	122	406	24	367
5-GRAM/2-SLM+2-GRAM/4-SLM+5-GRAM/PLSA[1]	151	425	33	310

In Appendix B, we give examples of "perfect" sentences, "only semantically correct" sentences, and "only grammatically correct" sentences.

7. Conclusion and Future Work

We have built a powerful large-scale distributed composite language model which integrates well-known n-gram, SLM, and PLSA models under the directed MRF paradigm. The composite language model has been trained by performing a convergent N-best list approximate EM algorithm and a follow-up EM algorithm to improve word prediction power on corpora up to a billion tokens, and stored on a supercomputer. We have achieved drastic perplexity reductions and obtained significantly better translation quality measured by the Bleu score and "readability" of translations in the task of re-ranking the N-best list from a state-of-the-art parsing-based MT system. As far as we know, this is the first work building a complex large-scale distributed language model with a principled approach that simultaneously exploits syntactic, semantic, and lexical regularities and is still more powerful than n-grams trained on a very large corpus with up to a billion tokens. It is reasonable to conjecture that composite language models can achieve drastic perplexity reduction and significantly better translation quality than n-gram when trained on Web-scale corpora that have trillions of tokens.

As stated in Wang et al. (2010, p. 45), "Since Banko and Brill's pioneering work almost a decade ago (Banko and Brill 2001), it has been widely observed that the effectiveness of statistical natural language processing (NLP) techniques is highly susceptible to the data size used to develop them. As empirical studies have repeatedly shown that simple algorithms can often outperform their more complicated counterparts in wide varieties of NLP applications with large data sets, many have come to believe that it is the size of data, not the sophistication of the algorithms, that ultimately play the central role in modern NLP (Norvig 2008)." It is true that 'the more the data, the better the result,' a dictum recently reiterated in a somewhat stronger form in Halevy, Norvig, and Pereira (2009), but care needs to be taken here. As we explained in the last paragraph of Section 6.2, after we increase the size of data, we should also increase the complexity of the model in order to achieve best results. For language modeling in particular, because the expressive power of simple n-grams is rather limited, it is worthwhile to exploit latent semantic information and syntactic structure that constrain the generation of natural language; this usually involves designing sophisticated algorithms. Of course, this implies that it takes a huge amount of resources to perform the computation. As cloud computing becomes the dominant platform for data management and information processing as utility computing, this will become feasible, affordable, and cheap.

The development of the large-scale distributed composite language model is in its infancy; we are planning to deepen our research and push this research in its limit. Specifically, we plan to integrate more advanced topic language models such as LDA (Blei, Ng, and Jordan 2003) and resort to a hierarchical non-parametric Bayesian model (Teh 2006; Teh and Jordan 2010) for smoothing fractional counts due to latent variables to handle the sparse data problem in Kneser-Ney's sense in a principled manner, thus constructing a family of large-scale distributed composite lexical, syntactic, and semantic language models. Finally we will put this family of composite language models into a phrased-based machine translation decoder (Koehn, Och, and Marcu 2003) that produces a lattice of alternative translations/transcriptions or a syntax-based

decoder (Chiang 2005, 2007) that produces a forest of alternatives (such integration would, in the exact case, reside in an extremely difficult complexity class, probably PSPACE-complete) to significantly improve the performance of the state-of-the-art machine translation systems.

Appendix A: An Example of Sentence Probability

We chose a document from the LDC English Gigaword corpus to show how sentence probability varies when computed by 5-gram, 5-gram/PLSA, and 5-gram/PLSA+4-SLM/PLSA. The document tag is ⟨XIN_ENG_20041126_0168.story⟩. This document's perplexity computed by 5-gram, 5-gram+PLSA, 5-gram+4-SLM+PLSA, 5-gram/PLSA, and 5-gram/PLSA+4-SLM/PLSA that are trained using 1.3 billion tokens corpus is 97, 93, 83, 71, and 64, respectively. We show the first four sentences below.

⟨s⟩ *cpc initiates education campaign to strengthen members ' wavering convictions* ⟨/s⟩ ⟨s⟩ *by zhao lei* ⟨/s⟩ ⟨s⟩ *beijing nov. 'nmbr xinhua the communist party of china cpc has decided to launch a mass internal educational campaign from january next year to prevent its members from wavering in their convictions* ⟨/s⟩ ⟨s⟩ *the decision aiming to keep the nature of the party members intact was made at the meeting of the political bureau of the cpc central committee on this oct. 'nmbr the cpc 's top power organ* ⟨/s⟩ · · · · · ·

We then list the word conditional probabilities given its document history for the fourth sentence. The first line is the fourth sentence; the second line (a) denotes the natural log value of the conditional word probabilities given its document history computed by 5-gram; the third line (b) denotes the natural log value of the conditional word probabilities given its document history computed by 5-gram+PLSA; the fourth line (c) denotes the natural log value of the conditional word probabilities given its document history computed by 5-gram+PLSA+4-SLM; the fifth line (d) denotes the natural log value of the conditional word probabilities given its document history computed by 5-gram/PLSA; and the sixth line (e) denotes the natural log value of the conditional word probabilities given its document history computed by 5-gram/PLSA+4-SLM/PLSA.

	the	*decision*	*aiming*	*to*	*keep*	*the*	*nature*	*of*	*the*
a.	−2.00317	−5.99654	−14.9793	−0.852055	−4.68269	−1.49193	−9.84554	−0.526566	−0.671103
b.	−2.05502	−6.08843	−13.2655	−0.950885	−4.78594	−1.56474	−9.81423	−0.6258	−0.761926
c.	−2.05416	−6.07556	−13.3486	−0.871798	−4.69523	−1.57311	−9.99731	−0.897362	−0.829652
d.	−1.72696	−5.65359	−14.2013	−0.99068	−5.43248	−1.65002	−7.6	−0.612751	−0.731037
e.	−1.80167	−5.73861	−14.5548	−0.893825	−5.05692	−1.60568	−7.92909	−0.751419	−0.755122

	party	*members*	*intact*	*was*	*made*	*at*	*the*	*meeting*	*of*
a.	−6.52337	−5.93013	−14.992	−5.5802	−5.91863	−3.47798	−1.0155	−3.77026	−3.11882
b.	−6.48382	−6.00924	−13.8132	−5.57218	−5.98123	−3.56856	−1.1003	−3.87003	−3.14354
c.	−6.48696	−5.81026	−8.11845	−3.04638	−2.21191	−2.80501	−1.12155	−3.85156	−2.3551
d.	−3.46383	−5.03999	−15.242	−5.27819	−4.73655	−3.03394	−0.69443	−3.23709	−3.40986
e.	−3.80075	−5.16911	−8.52597	−3.38567	−2.54778	−2.74127	−0.790644	−3.36195	−2.64652

	the	*political*	*bureau*	*of*	*the*	*cpc*	*central*	*committee*
a.	−0.619712	−5.91994	−1.36559	−0.17816	−0.217888	−1.55966	−0.282506	−0.110539
b.	−0.710967	−5.96757	−1.47083	−0.278998	−0.313708	−1.66454	−0.387673	−0.215632
c.	−0.636643	−6.0839	−1.43513	−0.6519	−0.634246	−2.10113	−0.504145	−0.216812
d.	−0.475928	−4.13345	−0.527685	−0.226433	−0.204276	−1.55903	−0.379722	−0.147238
e.	−0.475442	−4.43649	−0.702968	−0.427385	−0.388118	−1.79781	−0.42272	−0.136813

	on	*this*	*oct.*	*'nmbr*	*the*	*cpc*	*'s*	*top*	*power*
a.	−4.33953	−7.02792	−10.7495	−0.0380615	−3.87067	−9.93617	−3.54366	−4.19702	−7.6261
b.	−4.37441	−6.88172	−10.6397	−0.141938	−3.65821	−8.81816	−3.60823	−4.29886	−7.64586
c.	−3.57338	−6.86285	−10.9656	−0.131813	−3.8662	−8.85551	−3.42688	−4.28615	−7.82392
d.	−4.61674	−6.49064	−13.0595	−0.255452	−3.73302	−5.55244	−3.60481	−3.97708	−7.85289
e.	−3.85647	−6.61406	−12.5666	−0.178075	−3.92356	−5.90511	−3.46416	−4.03158	−7.91198

	organ	⟨/*s*⟩
a.	−5.97561	−2.62716
b.	−6.08022	−2.67444
c.	−6.01553	−2.65078
d.	−4.84265	−2.76932
e.	−5.05393	−2.70787

The conditional probability of the word(s) *party* or *political bureau* given document history computed by 5-gram/PLSA or 5-gram/PLSA+4-SLM/PLSA is significantly boosted due to the appearance of semantic related words such as *cpc* and *communist party* in the previous sentences, this clearly shows that the composite language models (5-gram/PLSA and 5-gram/PLSA+4-SLM/PLSA) trigger long-span document-level discourse topics to influence word prediction. In contrast, there is no effect when using linear combination models (i.e., 5-gram+PLSA and 5-gram+4-SLM+PLSA). Similarly, the conditional probability of the words *was made* (or the word *intact*) given document history computed by 5-gram/PLSA+4-SLM/PLSA is significantly boosted due the appearance of the grammatical headword *decision* (or *keep*) in the same sentence, this clearly shows that the composite language model (5-gram/PLSA+4-SLM/PLSA) exploits sentence level syntactic structure to influence word prediction. In this case, the n-gram has to increase its order to 11 or 8. The linear combination model 5-gram+4-SLM+PLSA is quite effective, although it has negative impact for the prediction of function words such as *of the* after the word(s) *natural* or *political bureau*.

Table 13 shows the statistics when n-grams are the same as the SLM's WORD-PREDICTOR in the most likely parse structure of each sentence in training corpora. Whenever the n-grams are not the same as SLM's WORD-PREDICTOR, the SLM component will be effective to furnish sentence-level long-range grammatical information.

This example and Table 13 clearly demonstrate that an n-gram alone is not able to achieve a similar effect to. SLM and PLSA even using Web-scale data, and the directed MRF paradigm effectively synergizes n-gram, m-SLM, and PLSA in a complementary, supplementary, and coherent way to form a powerful language model for word prediction of natural language.

Table 13
Statistics when n-grams are the same as SLM's WORD-PREDICTOR in the most likely parse structure of each sentence in training corpora.

CORPUS	$w_{-2}^{-1} = h_{-2}^{-1}$	$w_{-3}^{-1} = h_{-3}^{-1}$	$w_{-4}^{-1} = h_{-4}^{-1}$
44 M	57%	46%	38%
230 M	59%	46%	38%
1.3 B	55%	48%	43%

Appendix B: Examples of Translation Results

In the following, we give examples of "perfect" sentences, "only semantically correct" sentences, and "only grammatically correct" sentences, where the digit numbers are the sentence number in the *N*-best list from Hiero (a) denotes the reference sentence, (b) denotes the result provided by the composite language model, and (c) denotes the result provided by 5-gram.

A few examples of "perfect" sentences provided by the composite language model:

—512—

a. Sri Lanka's Prime Minister Calls on the People to Work together for Permanent Peace

b. Sri Lanka prime minister called on national common efforts to achieve lasting peace

c. Sri Lanka prime minister called on the national common achieve lasting peace

—54—

a. Wilner said the maximum penalty for securities fraud is 10 years imprisonment. However, the sentence is expected to be "significantly shorter" under the plea deal.

b. wiener, said securities fraud charges could be sentenced to 10 years' imprisonment, according to pleaded guilty mitigation, the sentence is "shorten".

c. wiener, sentenced to 10 years' imprisonment maximum securities fraud charges, according to pleaded guilty mitigation, the sentence is "shorten".

—206—

a. He said at a press conference in Doha, capital of Qarta, that if the United States "attacks Iraq, it may trigger a global disaster."

b. his press conference in doha, capital of qatar, said "if the united states attacks iraq, it will trigger a world disaster".

c. his press conference in doha, capital of qatar, said that the united states attacks iraq, "if it will trigger a world disaster".

—249—

a. Some Areas in Northwest Australia Face floods

b. floods in some areas in the northwest australia

c. australia northwest part of floods

A few examples of "only grammatically correct" sentences provided by the composite language model:

—458—

a. Sutiyoso said that gardens and flower beds would reduce the impression that the US embassy is a fort.

b. szudy about woven said that garden landscape could reduce the us embassy to a fortress.

c. szudy over so that garden landscape can reduce the u.s. embassy to a fortress.

—676—

a. He said that during last Christmas and the New Year, mainland tourists' spending accounted for 30

b. during christmas last year, he said, the mainland visitors spending will account for a three to four percent of the kaneyuki business and become the major consumer of the industry.

c. last year, he said, mainland visitors during the christmas spending for the kaneyuki 3 to 4 percent of the business, has become the major consumption.

A few examples of "only semantically correct" sentences provided by the composite language model:

—507—

a. The famous historic city of Cologne also narrowly escaped the disaster in the heavy rains.

b. cologne, a famous historical city also escaped unscathed in the heavy rain.

c. cologne, a famous historical city in heavy rain, escaped unscathed.

—416—

a. However, he insisted on the timetable laid down by Bush. That is UN only has "weeks but not months" to try to disarm Iraq peacefully and it would be military action thereafter.

b. however, he insists the bush timetable, the united nations is "weeks rather than months" to urge iraq to the peace disarm, then we will take military action.

c. however, he insists that the bush timetable, the only "weeks rather than months" to urge iraq to the peace disarm, she went on to take military action.

—787—

a. France circulated its proposals in the form of "a non-paper."

b. france is to distribute their proposals in the form of "non - paper."

c. france is the form of "non - paper" distribute their proposals.

—313—

a. In China, three-quarters of the 1.3 billion population were reported to have celebrated the New Year by watching television.

b. 1.3 billion population in china, according to reports, 3 / 4 is to watch tv celebrate lunar new year.

c. 1.3 billion population in china, according to reports, 3 / 4 is to celebrate televisions.

Acknowledgments

We would like to dedicate this work to the memory of Fred Jelinek, who passed away while we were finalizing this manuscript. Fred Jelinek laid the foundation for modern speech recognition and text translation technology. His work has greatly influenced us. This research is supported by the National Science Foundation under grant IIS RI-small 0812483, a Google research award, and Air Force Office of Scientific Research under grant FA9550-10-1-0335. We would like to thank the Ohio Supercomputer Center for an allocation of computing time to make this research possible; Ciprian Chelba for providing the SLM code, answering many questions regarding SLM, and consulting on various aspects of the work; Ying Zhang and Philip Resnik for providing the 1,000-best list from Hiero for re-ranking in machine translation; Peng Xu for suggesting to look at the conditional probability of a word given its document history to make the perplexity result much more convincing. Finally we would also like to thank the reviewers, who made a number of invaluable suggestions about the writing of the paper and pointed out many weaknesses in our original manuscript.

References

Aho, A. and J. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, Volume 1: Parsing. Prentice-Hall, Upper Saddle River, NJ.

Amari, S. and H. Nagaoka. 2000. *Methods of Information Geometry*. Translations of Mathematical Monographs; v. 191, American Mathematical Society, Providence, RI.

Bahl, L., J. Baker, F. Jelinek, and R. Mercer. 1977. Perplexity: A measure of difficulty of speech recognition tasks. *94th Meeting of the Acoustical Society of America*, 62:S63, Supplement 1, Miami, FL.

Baker, J. 1979. Trainable grammars for speech recognition. *The 97th Meeting of the Acoustical Society of America*, 547–550, Cambridge, MA.

Banko, M. and E. Brill. 2001. Mitigating the paucity-of-data problem: Exploring the effect of training corpus size on classifier performance for natural language processing. *Proceedings of the 1st International Conference on Human Language Technology Research* (HLT), 1–5, Strodsburg, PA.

Barron, A. and C. Sheu. 1991. Approximation of density functions by sequences of exponential families. *Annals of Statistics*, 19:1347–1369.

Bellegarda, J. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of IEEE*, 88(8):1279–1296.

Bellegarda, J. 2001. Robustness in statistical language modeling: Review and perspectives. In J. Junqua and G. van Noods, editors, *Robustness in Language and Speech Technology*, pages 101–121. Kluwer Academic Publishers, Dordrecht.

Bellegarda, J. 2003. Statistical language model adaptation: Review and perspectives. *Speech Communication*, 42:93–108.

Benedí, J. and J. Sánchez. 2005. Estimation of stochastic context-free grammars and their use as language models. *Computer Speech and Language*, 19(3):249–274.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Berger, A., S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Bilmes, J. and K. Kirchhoff. 2003. Factored language models and generalized parallel backoff. *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies* (NAACL-HLT), 4–6, Edmonton, Alberta, Canada.

Blei, D., A. Ng, and M. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Brants, T., A. Popat, P. Xu, F. Och, and J. Dean. 2007. Large language models in machine translation. *The 2007 Conference on Empirical Methods in Natural Language Processing* (EMNLP), 858–867, Prague, Czech Republic.

Charniak, E. 2001. Immediate-head parsing for language models. *The 39th Annual Conference on Association of Computational Linguistics* (ACL), 124–131, Toulouse, France.

Charniak, E., K. Knight, and K. Yamada. 2003. Syntax-based language models for statistical machine translation. *MT Summit IX, International Association for Machine Translation*, 40–46, New Orleans, LA.

Chelba, C. 2000. *Exploiting Syntactic Structure for Natural Language Modeling*.

Ph.D. dissertation, The Johns Hopkins University, Baltimore, MD.
Chelba, C. and F. Jelinek. 1998. Exploiting syntactic structure for language modeling. *The 36th Annual Conference on Association of Computational Linguistics* (ACL), 225–231, Montreal, Quebec, Canada.
Chelba, C. and F. Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.
Chelba, C., J. Schalkwyk, T. Brants, V. Ha, B. Harb, W. Neveitt, C. Parada, and P. Xu. 2010. Query language modeling for voice search. *Proceedings of the 2010 IEEE Workshop on Spoken Language Technology* (SLT), 127–132, Berkeley, CA.
Chen, S. and J. Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4): 319–358.
Chiang, D. 2005. A hierarchical phrase-based model for statistical machine translation. *The 43th Annual Conference on Association of Computational Linguistics* (ACL), 263–270, Ann Arbor, MI.
Chiang, D. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
Dean, J. and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. *The Sixth Symposium on Operating Systems Design and Implementation* (OSDI), 137–150, San Francisco, CA.
Della Pietra, S., V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, and L. Ures. 1994. Inference and estimation of a long-range trigram model. *Second International Colloquium on Grammatical Inference and Applications* (ICGI), pages 78–92, Springer-Verlag, Alicante, Spain.
Della Pietra, S., V. Della Pietra and J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.
Dempster, A., N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1–38.
Emami, A., K. Papineni, and J. Sorensen. 2007. Large-scale distributed language modeling. *The 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), pages 37–40, Honolulu, HI.
Gildea, D. and T. Hofmann. 1999. Topic-based language models using EM. *The 6th European Conference on Speech Communication and Technology* (EUROSPEECH), pages 2167–2170.
Goodman, J. 2001. A bit of progress in language modeling. *Computer Speech and Language*, 15(4):403–434.
Halevy, A., P. Norvig, and F. Pereira. 2009. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12.
Hastie, T., R. Tibshirani and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd edition. Springer, Berlin.
Hofmann, T. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196.
Jelinek, F. 1991. Up from trigrams! The struggle for improved language models. *Second European Conference on Speech Communication and Technology* (EUROSPEECH), pages 1037–1040, Genove, Italy.
Jelinek, F. 1998. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA.
Jelinek, F. 2004. Stochastic analysis of structured language modeling. In M. Johnson, S. Khudanpur, M. Ostendorf, and R. Rosenfeld, editors, *Mathematical Foundations of Speech and Language Processing*, pages 37–72, Springer-Verlag, Berlin.
Jelinek, F. 2009. The dawn of statistical ASR and MT. *Computational Linguistics*, 35(4):483–494.
Jelinek, F. and C. Chelba. 1999. Putting language into language modeling. *Sixth European Conference on Speech Communication and Technology* (EUROSPEECH), Keynote Paper 1, Budapest, Hungary.
Jelinek, F. and R. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In E. Gelsema and L. Kanal, editors, *Pattern Recognition in Practice*, pages 381–397. North Holland Publishers, Amsterdam.
Jurafsky, D. and J. Martin. 2008. *Speech and Language Processing*, 2nd edition. Prentice Hall, Upper Saddle River, NJ.
Khudanpur, S. and J. Wu. 2000. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech and Language*, 14(4):355–372.
Kneser, R. and H. Ney. 1995. Improved backing-off for *m*-gram language modeling. *The 20th IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), 181–184, Detroit, MI.
Koehn, P. 2004. Statistical significance tests for machine translation evaluation. *The*

2004 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 388–395, Barcelona, Spain.
Koehn, P., F. Och, and D. Marcu. 2003. Statistical phrase-based translation. *The Human Language Technology Conference* (HLT), pages 48–54, Edmonton, Alberta, Canada.
Lari, K. and S. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.
Lau, R., R. Rosenfeld, and S. Roukos. 1993. Trigger-based language models: A maximum entropy approach. *The 18th IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), II:45–48, Minneapolis, MN.
Lauritzen, S. 1996. *Graphical Models*. Oxford University Press.
Lavie, A., D. Yarowsky, K. Knight, C. Callison-Burch, N. Habash, and T. Mitamura. 2006. MINDS Workshops Machine Translation Working Group Final Report. Available at `http://www-nlpir.nist.gov/MINDS/FINAL/MT.web.pdf`.
Lin, J. and C. Dyer. 2010. *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool Publishers.
Mark, K., M. Miller, and U. Grenander. 1996. Constrained stochastic language models, In S. Levinson and L. Shepp, editors, *Image Models and Their Speech Model Cousins*, pages 131–137, Springer-Verlag, Berlin.
McAllester, D., M. Collins, and F. Pereira. 2004. Case-factor diagrams for structured probabilistic modeling. *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* (UAI), pages 382–391, Banff, Canada.
Norvig, P. 2008. Statistical learning as the ultimate agile development tool. *ACM 17th Conference on Information and Knowledge Management* (CIKM) Industry Event, Napa Valley, CA.
Och, F. 2003. Minimum error rate training in statistical machine translation. *The 41th Annual Meeting of the Association for Computational Linguistics* (ACL), pages 160–167, Sapporo, Japan.
Och, F. 2005. Statistical machine translation: Foundations and recent advances. *Presentation at MT-Summit*. `http://www.mt-archive.info/MTS-2005-och.pdf`
Papineni, K., S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. *The 40th Annual Meeting of the Association for Computational Linguistics* (ACL), pages 311–318, Philadelphia, PA.
Pereira, F. 2000. Formal grammar and information theory: Together again? *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, 358(1769):1239–1253.
Roark, B. 2001. Probabilistic top–down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.
Rosenfeld, R. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10(2):187–228.
Rosenfeld, R. 2000a. Two decades of statistical language modeling: Where do we go from here? *Proceedings of IEEE*, 88(8):1270–1278.
Rosenfeld, R. 2000b. Incorporating linguistic structure into statistical language models. *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, 358(1769):1311–1324.
Rosenfeld, R., S. Chen, and X. Zhu. 2001. Whole-sentence exponential language models: A vehicle for linguistic-statistical integration. *Computer Speech and Language*, 15(1): 55–73.
Russell, S. and P. Norvig. 2010. *Artificial Intelligence: A Modern Approach*, 3rd edition. Prentice Hall, Upper Saddle River, NJ.
Saul, L. and F. Pereira. 1997. Aggregate and mixed-order Markov models for statistical language processing. *The Second Conference on Empirical Methods in Natural Language Processing* (EMNLP), 81–89, Providence, RI.
Teh, Y. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. *The 44th Annual Conference of the Association of Computational Linguistics* (ACL), 985–992, Sydney, Australia.
Teh, Y. and M. Jordan. 2010. Hierarchical Bayesian nonparametric models with applications. In N. Hjort, C. Holmes, P. Mueller, and S. Walker, editors, *Bayesian Nonparametrics: Principles and Practice*, pages 158–207, Cambridge University Press.
Van Uytsel, D. and D. Compernolle. 2005. Language modeling with probabilistic left corner parsing. *Computer Speech and Language*, 19(2):171–204.
Vapnik, V. 1998. *Statistical Learning Theory*. Springer, Berlin.

Wallach, H. 2006. Topic modeling: Beyond bag-of-words. *The 23rd International Conference on Machine Learning* (ICML), 977–984, Pittsburgh, PA.

Wang, S., R. Greiner, and S. Wang. 2009. Consistency and generalization bounds for maximum entropy density estimation. *Manuscript*.

Wang, W. and M. Harper. 2002. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. *The 2002 Conference on Empirical Methods in Natural Language Processing* (EMNLP), pages 238–247, Philadelphia, PA.

Wang, S., D. Schuurmans, F. Peng, and Y. Zhao. 2005a. Combining statistical language models via the latent maximum entropy principle. *Machine Learning Journal: Special Issue on Learning in Speech and Language Technologies*, 60:229–250.

Wang, S., D. Schuurmans, and Y. Zhao. 2012. The latent maximum entropy principle. *ACM Transactions on Knowledge Discovery from Data* (TKDD) to appear. In Press.

Wang, K., C. Thrasher, E. Viegas, X. Li, and P. Hsu. 2010. An overview of Microsoft web N-gram corpus and applications. *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies* (NAACL-HLT): Demonstration Session, pages 45–48, Los Angeles, CA.

Wang, S., S. Wang, L. Cheng, R. Greiner, and D. Schuurmans. 2006. Stochastic analysis of lexical and semantic enhanced structural language model. *The 8th International Colloquium on Grammatical Inference* (ICGI), pages 97–111, Tokyo, Japan.

Wang, S., S. Wang, R. Greiner, D. Schuurmans, and L. Cheng. 2005b. Exploiting syntactic, semantic and lexical regularities in language modeling via directed Markov random fields. *The 22nd International Conference on Machine Learning* (ICML), 953–960, Bonn, Germany.

Wu, C. 1983. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11:95–103.

Yamada, K. and K. Knight. 2001. A syntax-based statistical translation model. *Proceedings of the 39th Annual Conference of the Association of Computational Linguistics* (ACL), 1067–1074, Toulouse, France.

Zangwill, W. 1969. *Nonlinear Programming: A Unified Approach*. Prentice-Hall, Upper Saddle River, NJ.

Zhang, Y. 2008. *Structured Language Models for Statistical Machine Translation*. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA.

Zhang, Y., A. Hildebrand, and S. Vogel. 2006. Distributed language modeling for N-best list re-ranking. *The 2006 Conference on Empirical Methods in Natural Language Processing* (EMNLP), 216–223, Sydney, Australia.

Zhang, Y., S. Vogel, A. Emami, K. Papineni, J. Sorensen, and J. Quinn. 2011. Distributed language modeling. In Joseph Olive, Caitlin Christianson, and John McCary, editors, *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*, Chapter 2.5.1, 252–270, Springer.