A Large Scale Distributed Syntactic, Semantic and Lexical Language Model for Machine Translation

Ming Tan Wenli Zhou Lei Zheng Shaojun Wang
Kno.e.sis Center
Department of Computer Science and Engineering
Wright State University
Dayton, OH 45435, USA
{tan.6,zhou.23,lei.zheng,shaojun.wang}@wright.edu

Abstract

This paper presents an attempt at building a large scale distributed composite language model that simultaneously accounts for local word lexical information, mid-range sentence syntactic structure, and long-span document semantic content under a directed Markov random field paradigm. The composite language model has been trained by performing a convergent N-best list approximate EM algorithm that has linear time complexity 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" 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 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. Research groups (Brants et al., 2007; Zhang, 2008) have shown that using an immense distributed computing paradigm, up to 6-grams can be trained on up to billions and trillions of words, yielding consistent system improvements, but Zhang (2008) did not observe much improvement beyond 6-grams. Although the Markov chains are efficient at encoding local word interactions, the n-gram model clearly ignores the rich syntactic and semantic structures that constrain natural languages. As the machine translation (MT) working groups stated on page 3 of their final report (Lavie et al., 2006), "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."

Wang et al. (2006) integrated n-gram, structured language model (SLM) (Chelba and Jelinek, 2000) and probabilistic latent semantic analysis (PLSA) (Hofmann, 2001) under the directed MRF framework (Wang et al., 2005) and studied the stochastic properties for the composite language model. They derived a *generalized inside-outside* algorithm to train the composite language model from a general EM (Dempster et al., 1977) by following Jelinek's ingenious definition of the inside and outside probabilities for SLM (Jelinek, 2004) with 6th order of sentence length time complexity. Unfortunately, there are no experimental results reported.

In this paper, we study the same composite language model. Instead of using the 6th order generalized inside-outside algorithm proposed in (Wang et al., 2006), we train this composite model by a convergent N-best list approximate EM algorithm that has linear time complexity and a follow-up EM algorithm to improve word prediction power. 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, we obtain drastic perplexity reductions. Finally, we ap-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 201–210,
Portland, Oregon, June 19-24, 2011.

ply our language models to the task of re-ranking the N-best list from Hiero (Chiang, 2005; Chiang, 2007), a state-of-the-art parsing-based MT system, we achieve significantly better translation quality measured by the BLEU score and "readability".

2 Composite language model

The n-gram language model is essentially a word predictor that given its entire document history it predicts next word w_{k+1} based on the last n-1 words with probability $p(w_{k+1}|w_{k-n+2}^{k})$ where $w_{k-n+2}^{k} = w_{k-n+2}, \cdots, w_k$.

The SLM (Chelba and Jelinek, 1998; Chelba and Jelinek, 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 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 $<s>$ and appended the sentence end marker $</s>$ so that $w_0 =<s>$ and $w_{n+1} =</s>$. 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 $W_k T_k$ the word-parse k-prefix. A word-parse k-prefix has a set of exposed heads $h_{-m}, \cdots, h_{-1}$, with each head being a pair (headword, non-terminal label), or in the case of a root-only tree (word, POS tag). An *m-th order SLM* (m-SLM) has three operators to generate a sentence: WORD-PREDICTOR predicts the next word w_{k+1} based on the m left-most 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} to the next word w_{k+1} based on the next word w_{k+1} and the POS tags of the m left-most exposed headwords h_{-m}^{-1} in the word-parse k-prefix with probability $p(t_{k+1}|w_{k+1}, h_{-m}.\text{tag}, \cdots, h_{-1}.\text{tag})$; the CONSTRUCTOR builds the partial parse T_k from T_{k-1}, w_k, and t_k 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}. Once the CONSTRUCTOR hits NULL, it passes control to the WORD-PREDICTOR. See detailed description in (Chelba and Jelinek, 2000).

A PLSA model (Hofmann, 2001) is a generative probabilistic model of word-document co-occurrences using the bag-of-words assumption described as follows: (i) choose a document d with probability $p(d)$; (ii) SEMANTIZER: select a semantic class g with probability $p(g|d)$; and (iii) WORD-PREDICTOR: pick a word w with probability $p(w|g)$. Since only one pair of (d, w) is being observed, as a result, the joint probability model is a mixture of log-linear model with the expression $p(d, w) = p(d) \sum_g p(w|g)p(g|d)$. Typically, the number of documents and vocabulary size are much larger than the size of latent semantic class variables. Thus, latent semantic class variables function as bottleneck variables to constrain word occurrences in documents.

When combining n-gram, m order SLM and PLSA models together to build a composite generative language model under the directed MRF paradigm (Wang et al., 2005; Wang et al., 2006), the TAGGER and CONSTRUCTOR in SLM and SEMANTIZER in PLSA remain unchanged; however the WORD-PREDICTORs in n-gram, m-SLM and PLSA are combined to form a stronger WORD-PREDICTOR that generates the next word, w_{k+1}, not only depending on the m left-most 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 $p(w_{k+1}|w_{k-n+2}^{k} h_{-m}^{-1} g_{k+1})$. The resulting composite language model has an even more complex dependency structure but with more expressive power than the original SLM. Figure 1 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 directed MRF model (Wang et al., 2006) with local normalization constraints for the parameters of each model component, WORD-PREDICTOR, TAGGER, CONSTRUCTOR, SEMANTIZER, i.e., $\sum_{w \in \mathcal{V}} p(w|w_{-n+1}^{-1} h_{-m}^{-1} g) = 1, \sum_{t \in \mathcal{O}} p(t|w h_{-m}^{-1}.\text{tag}) = 1, \sum_{a \in \mathcal{A}} p(a|h_{-m}^{-1}) = 1, \sum_{g \in \mathcal{G}} p(g|d) = 1$.

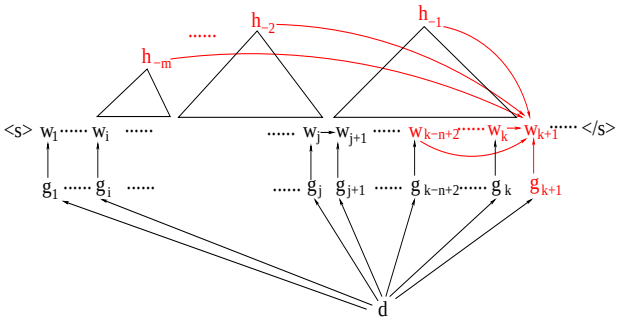

Figure 1: *A composite n-gram/m-SLM/PLSA language model where the hidden information is the parse tree T and semantic content g. The WORD-PREDICTOR generates the next word w_{k+1} with probability $p(w_{k+1}|w_{k-n+2}^{k}h_{-m}^{-1}g_{k+1})$ instead of $p(w_{k+1}|w_{k-n+2}^{k})$, $p(w_{k+1}|h_{-m}^{-1})$ and $p(w_{k+1}|g_{k+1})$ respectively.*

3 Training algorithm

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

$$\mathcal{L}(\mathcal{D},p)=\prod_{d\in\mathcal{D}}\left(\prod_{l}\left(\sum_{G^l}\left(\sum_{T^l}P_p(W^l,T^l,G^l|d)\right)\right)\right) \quad (1)$$

where (W^l,T^l,G^l,d) denote the joint sequence of the lth sentence W^l with its parse tree 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

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

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, $\#(w_{-n+1}^{-1}wh_{-m}^{-1}g,W^l,T^l,G^l,d)$ is the count of n-grams, its m most recent 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, $\#(twh_{-m}^{-1}.\text{tag},W^l,T^l,d)$ is the count of tag t predicted by word w and the tags of m most recent exposed headwords in parse tree T^l of the lth sentence W^l in document d, and finally $\#(ah_{-m}^{-1},W^l,T^l,d)$ 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.

The objective of maximum likelihood estimation is to maximize the likelihood $\mathcal{L}(\mathcal{D},p)$ 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 exponential with sentence length, Wang et al. (2006) have derived a generalized inside-outside algorithm by applying the standard EM algorithm. However, the complexity of this algorithm is 6th order of sentence length, 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, 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,

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

where $\mathcal{T'}^l_N$ is a set of N parse trees for sentence W^l in document d and $||\cdot||$ 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 EM algorithm to estimate model

parameters that maximizes N-best-list likelihood of the training corpus $\mathcal{D}$,

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

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 new update for p.

Iterate steps (1) and (2) until the convergence of the N-best-list likelihood. Due to space constraints, we omit the proof of the convergence of the N-best list approximate EM algorithm which uses Zangwill's global convergence theorem (Zangwill, 1969).

N-best list search strategy: 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, 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 $\log(\sum_{G_k} P_p(W_k, T_k, G_k|d))$ score with the highest on top, where $P_p(W_k, T_k, G_k|d)$ is the joint probability of prefix $W_k = w_0, \cdots, w_k$ with its parse structure T_k and semantic annotation string $G_k = g_1, \cdots, g_k$ in a document d. A stack vector consists of the ordered set of stacks containing partial parses with the same number of WORD-PREDICTOR operations but different number of CONSTRUCTOR operations. In WORD-PREDICTOR and TAGGER operations, some hypotheses are discarded due to the maximum number of hypotheses the stack can contain at any given time. In 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.

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

In E-step, we compute the expected count of each model parameter over sentence W^l in document d in the training corpus $\mathcal{D}$. For the WORD-PREDICTOR and the SEMANTIZER, the number of possible semantic annotation sequences is exponential, we use forward-backward recursive formulas that are similar to those in hidden Markov models to compute the expected counts. We define the forward vector $\alpha^l(g|d)$ to be

$$\alpha^l_{k+1}(g|d) = \sum_{G^l_k} P_p(W^l_k, T^l_k, w^k_{k-n+2} w_{k+1} h^{-1}_{-m} g, G^l_k | d)$$

that can be recursively computed in a forward manner, where W^l_k is the word k-prefix for sentence W^l, T^l_k is the parse for k-prefix. We define backward vector $\beta^l(g|d)$ to be

$$\beta^l_{k+1}(g|d) = \sum_{G^l_{k+1,\cdot}} P_p(W^l_{k+1,\cdot}, T^l_{k+1,\cdot}, G^l_{k+1,\cdot} | w^k_{k-n+2} w_{k+1} h^{-1}_{-m} g, d)$$

that can be computed in a backward manner, here $W^l_{k+1,\cdot}$ is the subsequence after k+1th word in sentence W^l, $T^l_{k+1,\cdot}$ is the incremental parse structure after the parse structure T^l_{k+1} of word k+1-prefix W^l_{k+1} that generates parse tree T^l, $G^l_{k+1,\cdot}$ is the semantic subsequence in G^l relevant to $W^l_{k+1,\cdot}$. Then, the expected count of $w^{-1}_{-n+1} w h^{-1}_{-m} g$ for the WORD-PREDICTOR on sentence W^l in document d is

$$\sum_{G^l} P_p(T^l, G^l | W^l, d) \#(w^{-1}_{-n+1} w h^{-1}_{-m} g, W^l, T^l, G^l, d)$$
$$= \sum_{l} \sum_{k} \alpha^l_{k+1}(g|d) \beta^l_{k+1}(g|d) p(g|d)$$
$$\delta(w^k_{k-n+2} w_{k+1} h^{-1}_{-m} g_{k+1} = w^{-1}_{-n+1} w h^{-1}_{-m} g) / P_p(W^l | d)$$

where $\delta(\cdot)$ is an indicator function and the expected count of g for the SEMANTIZER on sentence W^l in document d is

$$\sum_{G^l} P_p(T^l, G^l | W^l, d) \#(g, W^l, G^l, d)$$
$$= \sum_{k=0}^{j-1} \alpha^l_{k+1}(g|d) \beta^l_{k+1}(g|d) p(g|d) / P_p(W^l | d)$$

For the TAGGER and the CONSTRUCTOR, the expected count of each event of $t w h^{-1}_{-m}.\mathrm{tag}$ and $a h^{-1}_{-m}$ over parse T^l of sentence W^l in

document d is the real count appeared in parse tree T^l of sentence W^l in document d times the conditional distribution $P_p(T^l|W^l,d) = P_p(T^l,W^l|d)/\sum_{T^l\in\mathcal{T}^l} P_p(T^l,W^l|d)$ respectively.

In M-step, the recursive linear interpolation scheme (Jelinek and Mercer, 1981) 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, 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 $k = 0,\cdots,n$ as explained in (Chelba and Jelinek, 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 original recursive mixing scheme (Jelinek and Mercer, 1981) and form a lattice to handle the situation where the context is a mixture of Markov chains.

3.2 Follow-up EM

As explained in (Chelba and Jelinek, 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 WORD-PREDICTOR using the algorithm below.

The *language model* probability assignment for the word at position k+1 in the input sentence of document d can be computed as

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

where $P_p(T_k|W_k,d) = \frac{\sum_{G_k} P_p(W_k,T_k,G_k|d)}{\sum_{T_k\in Z_k}\sum_{G_k} P_p(W_k,T_k,G_k|d)}$ and 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.

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

$$\tilde{\mathcal{L}}(\mathcal{D},p) = \prod_{d\in\mathcal{D}}\prod_{l}\Big(\sum_{k} P_p(w_{k+1}^{(l)}|W_k^l,d)\Big) \quad (3)$$

We employ 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 using EM again to maximize Equation (3) to improve the predictive power of WORD-PREDICTOR.

3.3 Distributed architecture

When using very large corpora to train our composite language model, both the data and the parameters can't be stored in a single machine, so we have to resort to distributed computing. The topic of large scale distributed language models is relatively new, and existing works are restricted to n-grams only (Brants et al., 2007; Emami et al., 2007; Zhang et al., 2006). Even though all 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 mapped and 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. and make it suitable to perform iterations of N-best list approximate EM algorithm, see Figure 2. The corpus is divided and loaded into a number of clients. We use a public available parser to parse the sentences in each client to get the initial counts for $w_{-n+1}^{-1}wh_{-m}^{-1}g$ etc., 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

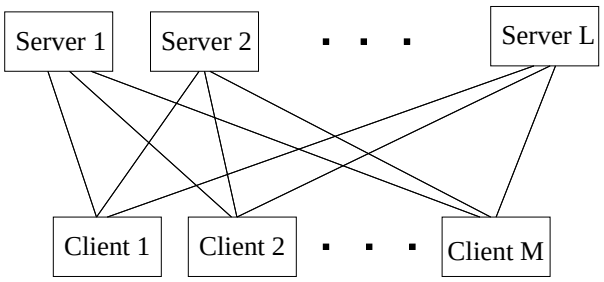

Figure 2: *Distributed architecture is essentially a MapReduce paradigm: clients store partitioned data and perform E-step: compute expected counts, this is Map; servers store parameters (counts) for M-step where counts of $w_{-n+1}^{-1}wh_{-m}^{-1}g$ are hashed by word w_{-1} (or h_{-1}) and its topic g to evenly distribute these model parameters into servers as much as possible, this is Reduce.*

of the servers by hashing through the word w_{-1} (or h_{-1}) and its topic g, finish 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 synchronous multi-stack search for each sentence to get the N-best list parse trees. Again, the expected count for a particular parameter of $w_{-n+1}^{-1}wh_{-m}^{-1}g$ at the clients are computed, thus we finish a Map part, then summed up and stored in one of the servers by hashing through the word w_{-1} (or h_{-1}) and its topic g, thus we finish the Reduce part. We repeat this procedure until convergence.

Similarly, we use a distributed architecture as in Figure 2 to perform the follow-up EM algorithm to re-estimate WORD-PREDICTOR.

4 Experimental results

We have trained our language models using three different training sets: one has 44 million tokens, another has 230 million tokens, and the other has 1.3 billion tokens. An independent test set which has 354 k tokens is chosen. The independent check data set used to determine the linear interpolation coefficients has 1.7 million tokens for the 44 million tokens training corpus, 13.7 million tokens for both 230 million and 1.3 billion tokens training corpora. All these data sets are taken from the LDC English Gigaword corpus with non-verbalized punctuation and we remove all punctuation. Table 1 gives the detailed information on how these data sets are chosen from the LDC English Gigaword corpus.

The vocabulary sizes in all three cases are:

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

	1.3 BILLION TOKENS 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 TOKENS TRAINING CORPUS
AFP	19940622.0336 ∼ 19961031.0797
APW	19941111.0001 ∼ 19960419.0765
NYT	19940701.0001 ∼ 19941130.0405
	44 MILLION TOKENS TRAINING CORPUS
AFP	19940601.0001 ∼ 19950721.0137
	13.7 MILLION TOKENS CHECK CORPUS
NYT	19950201.0001 ∼ 19950331.0494
	1.7 MILLION TOKENS CHECK CORPUS
AFP	19940512.0003 ∼ 19940531.0197
	354 K TOKENS TEST CORPUS
CNA	20041101.0006 ∼ 20041217.0009

Table 1: The corpora used in our experiments are selected from the LDC English Gigaword corpus and specified in this table, AFP, AFW, NYT, XIN and CNA denote the sections of the LDC English Gigaword corpus.

Similar to SLM (Chelba and Jelinek, 2000), after the parses undergo 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 (Northedge, 2005) to parse a large amount 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 Upenn Treebank used in (Chelba and Jelinek, 2000). For the 44 and 230 million tokens corpora, all sentences are automatically parsed and used to initialize model parameters, while for 1.3 billion tokens corpus, we parse the sentences from a portion of the corpus that

contain 230 million tokens, then use them to initialize model parameters. The parser at "openNLP" is trained by Upenn treebank with 1 million tokens and there is a mismatch between Upenn treebank and LDC English Gigaword corpus. Nevertheless, experimental results show that this approach is effective to provide initial values of model parameters.

As we have explained, the proposed EM algorithms can be naturally cast into a MapReduce framework, see more discussion in (Lin and Dyer, 2010). If we have access to a large cluster of machines with Hadoop installed that are powerful enough to process a billion tokens level corpus, we just need to specify a map function and a reduce function etc., Hadoop will automatically parallelize and execute programs written in this functional style. Unfortunately, we don't have this kind of resources available. Instead, we have access to a supercomputer at a supercomputer center with MPI installed that has more than 1000 core processors usable. Thus we implement our algorithms using C++ under MPI on the supercomputer, where we have to write C++ codes for Map part and Reduce part, and the MPI is used to take care of massage passing, scheduling, synchronization, etc. between clients and servers. This involves a fair amount of programming work, even though our implementation under MPI is not as reliable as under Hadoop but it is more efficient. We use up to 1000 core processors to train the composite language models for 1.3 billion tokens corpus where 900 core processors are used to store the parameters alone. We decide to use linearly smoothed trigram as the baseline model for 44 million token corpus, linearly smoothed 4-gram as the baseline model for 230 million token corpus, and linearly smoothed 5-gram as the baseline model for 1.3 billion token corpus. Model size is a big issue, we have to keep only a small set of topics due to the consideration in both computational time and resource demand. Table 2 shows the perplexity results and computation time of composite n-gram/PLSA language models that are trained on 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 rest are pruned. For composite 5-gram/PLSA model trained on 1.3 billion tokens corpus, 400 cores have to be used to keep top 5 most likely topics. For composite trigram/PLSA model trained on 44M tokens corpus, the computation time increases drastically with less than 5% percent perplexity improvement. So in the following experiments, we keep top 5 topics for each document from total 200 topics and all other 195 topics are pruned.

All composite language models are first trained by performing N-best list approximate EM algorithm until convergence, then EM algorithm for a second stage of parameter re-estimation for WORD-PREDICTOR and SEMANTIZER until convergence. We fix the size of topics in 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 3 shows comprehensive perplexity results for a variety of different models such as composite n-gram/m-SLM, n-gram/PLSA, m-SLM/PLSA, their linear combinations, etc., where we use online EM with 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. In Table 3, for 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 predictor's types by 85%. When we train the composite language on 1.3 billion tokens 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 1000 cores. In particular, for 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, and for 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, this significantly reduces the number of predictor's types by 85%. For 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 composite 4-SLM/PLSA model or its linear combination with models, we ignore all the tags and use only the words in the 4 head words. In this table, we have three items missing (marked by —), since the size of corresponding model is

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

Table 2: Perplexity (ppl) results and time consumed of composite n-gram/PLSA language model trained on three corpora when different numbers of most likely topics are kept for each document in PLSA.

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%	—	—	—	—

Table 3: Perplexity results for various language models on test corpus, where + denotes linear combination, / denotes composite model; n denotes the order of n-gram and m denotes the order of SLM; the topic nodes are pruned from 200 to 5.

too big to store in the supercomputer. 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 PLSA component, but when adding SLM component into n-gram/PLSA, there is a further 10% relative perplexity reduction.

We have applied our composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA language model that is trained by 1.3 billion word corpus for the task of re-ranking the N-best list in statistical machine translation. We used the same 1000-best list that is used by Zhang et al. (2006). This list was generated on 919 sentences from the MT03 Chinese-English evaluation set by Hiero (Chiang, 2005; Chiang, 2007), a state-of-the-art parsing-based translation model. Its decoder uses a trigram language model trained with modified Kneser-Ney smoothing (Kneser and Ney, 1995) on a 200 million tokens 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 partition the data into ten pieces, 9 pieces are used as training data to optimize the BLEU score (Papineni et al., 2002) by MERT (Och,

2003), a remaining single piece is used to re-rank the 1000-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. Table 4 shows the BLEU scores through 10-fold cross-validation. The composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA language model gives 1.57% BLEU score improvement over the baseline and 0.79% BLEU score improvement over the 5-gram. This is because there is not much diversity on the 1000-best list, and essentially only $20 \sim 30$ distinct sentences are there in the 1000-best list. 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, however, the BLEU score becomes significantly higher 37.09% when the n-gram is embedded directly into Hiero's one pass decoder, this is because there is not much diversity in the N-best list. It is expected that putting the our composite language into a one pass decoder of both phrase-based (Koehn et al., 2003) and parsing-based (Chiang, 2005; Chiang, 2007) MT systems should result in much improved BLEU scores.

SYSTEM MODEL	MEAN (%)
BASELINE	31.75
5-GRAM	32.53
5-GRAM/2-SLM+2-GRAM/4-SLM	32.87
5-GRAM/PLSA	33.01
5-GRAM/2-SLM+2-GRAM/4-SLM +5-GRAM/PLSA	33.32

Table 4: 10-fold cross-validation BLEU score results for the task of re-ranking the N-best list.

Besides reporting the BLEU scores, we look at the "readability" of translations similar to the study conducted by Charniak et al. (2003). The translations are sorted into four groups: good/bad syntax crossed with good/bad meaning by human judges, see Table 5. We find that many more sentences are perfect, many more are grammatically correct, and many more are semantically correct. The syntactic language model (Charniak, 2001; Charniak, 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 language model improves both significantly. Bear in mind that Charniak et al. (2003) integrated Charniak's language model with the syntax-based translation model Yamada and Knight proposed (2001) to rescore a tree-to-string translation forest, whereas we use only our language model for N-best list re-ranking. Also, in the same study in (Charniak, 2003), they 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 indicate 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" in the example of 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.

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	151	425	33	310

Table 5: Results of "readability" evaluation on 919 translated sentences, P: perfect, S: only semantically correct, G: only grammatically correct, W: wrong.

5 Conclusion

As far as we know, this is the first work of building a complex large scale distributed language model with a principled approach that is more powerful than n-grams when both trained on a very large corpus with up to a billion tokens. We believe our results still hold on web scale corpora that have trillion tokens, since the composite language model effectively encodes long range dependencies of natural language that n-gram is not viable to consider. Of course, this implies that we have to take a huge amount of resources to perform the computation, nevertheless this becomes feasible, affordable, and cheap in the era of cloud computing.

References

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

T. Brants et al.. 2007. Large language models in machine translation. *The 2007 Conference on Empirical Methods in Natural Language Processing* (EMNLP), 858-867.

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

E. Charniak, K. Knight and K. Yamada. 2003. Syntax-based language models for statistical machine translation. MT Summit IX., Intl. Assoc. for Machine Translation.

C. Chelba and F. Jelinek. 1998. Exploiting syntactic structure for language modeling. *The 36th Annual Conference on Association of Computational Linguistics* (ACL), 225-231.

C. Chelba and F. Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283-332.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. *The 43th Annual Conference on Association of Computational Linguistics* (ACL), 263-270.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201-228.

J. Dean and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. *Operating Systems Design and Implementation* (OSDI), 137-150.

A. Dempster, N. Laird and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1-38.

A. Emami, K. Papineni and J. Sorensen. 2007. Large-scale distributed language modeling. *The 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), IV:37-40.

T. Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177-196.

F. Jelinek and R. Mercer. 1981. Interpolated estimation of Markov source parameters from sparse data. *Pattern Recognition in Practice*, 381-397.

F. Jelinek and C. Chelba. 1999. Putting language into language modeling. *Sixth European Conference on Speech Communication and Technology* (EUROSPEECH), Keynote Paper 1.

F. Jelinek. 2004. Stochastic analysis of structured language modeling. *Mathematical Foundations of Speech and Language Processing*, 37-72, Springer-Verlag.

D. Jurafsky and J. Martin. 2008. *Speech and Language Processing*, 2nd Edition, Prentice Hall.

R. Kneser 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.

P. Koehn, F. Och and D. Marcu. 2003. Statistical phrase-based translation. *The Human Language Technology Conference* (HLT), 48-54.

S. Khudanpur 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.

A. Lavie et al. 2006. MINDS Workshops Machine Translation Working Group Final Report. http://www-nlpir.nist.gov/MINDS/FINAL/MT.web.pdf

J. Lin and C. Dyer. 2010. *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool Publishers.

R. Northedge. 2005. OpenNLP software http://www.codeproject.com/KB/recipes/englishparsing.aspx

F. Och. 2003. Minimum error rate training in statistical machine translation. *The 41th Annual meeting of the Association for Computational Linguistics* (ACL), 311-318.

K. Papineni, 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), 311-318.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249-276.

S. Wang et al. 2005. Exploiting syntactic, semantic and lexical regularities in language modeling via directed Markov random fields. *The 22nd International Conference on Machine Learning* (ICML), 953-960.

S. Wang et al. 2006. Stochastic analysis of lexical and semantic enhanced structural language model. *The 8th International Colloquium on Grammatical Inference* (ICGI), 97-111.

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

W. Zangwill. 1969. *Nonlinear Programming: A Unified Approach*. Prentice-Hall.

Y. Zhang, 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.

Y. Zhang, 2008. Structured language models for statistical machine translation. *Ph.D. dissertation*, CMU.