2013

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

Ming Tan
Wright State University

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

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

By

Ming Tan
M.S., Beijing Institute of Technology, 2006
B.S., Beijing Institute of Technology, 2004

2013
Wright State University
Dayton, Ohio 45435-0001

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

Oct. 14th, 2013

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Ming Tan ENTITLED A Large Scale Distributed Syntactic, Semantic and Lexical Language Model for Machine Translation BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Shaojun Wang, Ph.D.
Thesis Director

Arthur Ardeshir Goshtasby, Ph.D.
Director of Graduate Program

R. William Ayres, Ph.D.
Interim Dean,
Graduate School

Committee on
Final Examination

Shaojun Wang, Ph.D, adviser

Amit Sheth, Ph.D.

Krishnaprasad Thirunarayan, Ph.D.

Keke Chen, Ph.D.

Xinhui Zhang, Ph.D.

ABSTRACT

Ming Tan. Ph.D., Department of Computer Science and Engineering, Wright State University, 2013. A Large Scale Distributed Syntactic, Semantic and Lexical Language Model for Machine Translation.

The n-gram model is the most widely used language model (LM) in statistical machine translation system, due to its simplicity and scalability. However, it only encodes the local lexical relation between adjacent words and clearly ignores the rich syntactic and semantic structures of the natural languages. Attempting to increase the order of an n-gram to describe longer range dependencies in natural language immediately runs into the curse of dimensionality. Although previous researches tried to increase the order of n-gram on a large corpus, they did not see obvious improvement beyond 6-gram. Meanwhile, other LMs, such as syntactic language models and topic language models, tried to encode the long range dependencies from different perspectives of natural languages. But it is still an open question how to effectively combine those language models in order to capture multiple linguistic phenomena.

This dissertation presents a study at building a large scale distributed composite language model that is formed by seamlessly combining 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, the composite LM is distributed with client-server paradigm and

trained on corpora with up to a billion tokens. Also, the orders of the composite LM are increased up to 5-gram and 4-headword.

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.

Moreover, we propose an A^*-search-based lattice rescoring strategy in order to integrate the large scale distributed composite language model into a phrase-based machine translation system. Experiments show that the A^*-based lattice rescoring is more effective to show the predominance of the composite language model over the n-gram model than the traditional N-best list rescoring.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to express sincere thanks to my PhD superviser, Dr. Shaojun Wang, for his strong support and endless encouragement. I would like to extend my gratitude to Dr. Amit Sheth, Dr. Krishnaprasad Thirunarayan, Dr. Keke Chen and Dr. Xinhui Zhang for their time and advise while serving on my committee. I also would like to thank my colleague Tian Xia, Shaodan Zhai, Zhongliang Li, Raymond Kulhanek, as well as ex-colleague Wenli Zhou and Lei Zheng for their willingness to help and their co-operation.

1

Introduction

This work presents a large scale distributed composite language model that is formed by seamlessly integrating n-gram, structured language model and probabilistic latent semantic analysis under a directed Markov random field (MRF) paradigm to simultaneously account for local word lexical information, mid-range sentence syntactic structure, and long-span document semantic content.

1.1 Statistical machine translation

Machine translation (MT) is a topic of computational linguistics that investigates the use of software to translate text or speech from one natural language to another. Statistical machine translation (SMT) generates translations using statistical methods based on bilingual text corpora. In SMT, we assume we are translating from a foreign (source language) sentence $F = f_1, f_2, \cdots, f_m$ to English (target language) $E = e_1, e_2, \cdots, e'_m$. We apply the Bayesian noisy channel model to machine translation, such that we pretend that the foreign input F is a corrupted version of some English sentence E, and that our task is to discover the hidden sentence E that generates our observation sentence F. This procedure is illustrated in Figure 1.1. The best English translation is the hypothesis with the highest probability $P(E|F)$.

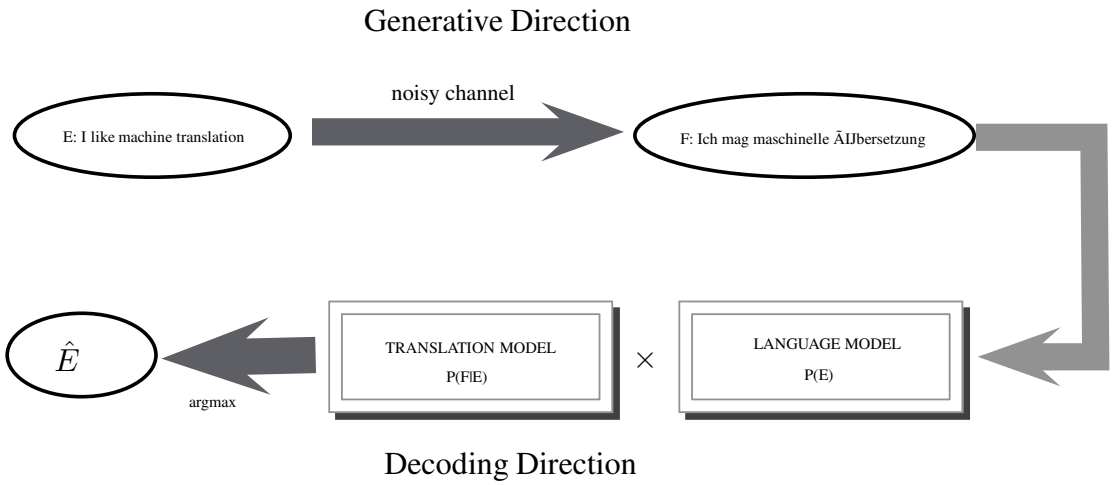

Figure 1.1: The noisy channel of statistical machine translation.

We can rewrite $P(E|F)$ via Bayes rule:

$$\begin{aligned} \hat{E} &= \arg\max_E P(E|F) \\ &= \arg\max_E \frac{P(F|E)P(E)}{P(F)} \\ &= \arg\max_E P(F|E)P(E) \end{aligned} \tag{1.1}$$

The denominator of Eq. 1.1, $P(F)$, is ignored, since we are choosing the best English sentence for a fixed foreign sentence F, so $P(F)$ is a constant. Therefore, there are basically two components in SMT: a *translation model* $P(F|E)$ and a *language model* $P(E)$. A *decoder* searches for the most probable translation E given the source sentence F. Intuitively speaking, the translation model describes the *faithfulness* of the translation, while the language model is a factor that influences the *fluency* of the translation result.

In this dissertation, we mainly focus on the language model (LM), and also discuss in detail how to exploit the proposed composite language model into a phrase-based MT system. The LM component is monolingual with the target language side, and so acquiring training data is relatively easy.

In recent years, this source-channel architecture has been generalized to a more

widely-used maximum entropy translation model [Och and Ney, 2002], in which the posterior probability $P(E|F)$ is directly modeled. In this framework, we have a set of features $\{h_m(E,F)\}_{m=1}^{M}$, which describe a variety of linguistic relationship between the source sentence and the translation hypothesis. For each feature, there exists a model parameter α_m, which is the weight of the given feature. The probability of an English translation is defined by,

$$P(E|F) = \frac{\exp\left(\sum_{m=1}^{M}\alpha_m h_m(E,F)\right)}{\sum_{E\in\mathcal{E}}\exp\left(\sum_{m=1}^{M}\alpha_m h_m(E,F)\right)}$$

where the denominator is a constant with respect to E. Then, we obtain the following translation result,

$$\begin{aligned}\hat{E} &= \arg\max_{E} P(E|F) \\ &= \arg\max_{E}\sum_{m=1}^{M}\alpha_m h_m(E,F) \qquad (1.2)\end{aligned}$$

Eq. 1.2 corresponds to the translation hypothesis with the maximum likelihood. The source-channel model is a special case of this framework, where there are only two features, $\log P(E)$ and $\log P(F|E)$, each of which has a weight of one.

Although the maximum entropy model does not require LMs to be a mandatory component, in most established commercial SMT system, LMs are still among the most important features. Intuitively, a LM evaluates the "fluency" of a translation result. Good LMs can help with finding a more natural sequential order of English words. For example,

$$P_{LM}(\textit{this is my home}) > P_{LM}(\textit{my this home is})$$

Moreover, good LMs can help the SMT system select an English word, more consistent with human habits. For example,

$$P_{LM}(\textit{I am going home}) > P_{LM}(\textit{I am going house})$$

If a translation hypothesis is more likely a human-spoken language, the LM score should be higher.

1.2 Language models

As shown in the previous section, language modeling is a key part of machine translation system. It can also be used in many natural language processing applications such as speech recognition, part-of-speech tagging, parsing and information retrieval. The LM estimates the values $P(W)$, where $W = w_1, w_2, \cdots, w_n$ is a English word sequence.

According to the Bayes theorem, we have following,

$$P(W) = \prod_{k=1}^{n} P(w_k|w_1, w_2, ..., w_{k-1})$$

$P(w_i|w_1, w_2, ..., w_{i-1})$ cannot be directed modeled due to the unbounded history $W_{k-1} = w_1 w_2, ..., w_{k-1}$, since the sentence can be very long. Therefore, LMs convert the history into an *equivalence class* determined by a function $\Phi(W_{k-1})$. That is,

$$P(W) = \prod_{k=1}^{n} P(w_k|\Phi(W_{k-1}))$$

The Markov chain (n-gram) source models, which predict each word on the basis of previous n-1 words,

$$\Phi(W_{k-1}) = w_{k-1} w_{k-2}, ..., w_{k-n+1} = w_{k-n+1}^{k-1}$$

n-gram LMs have been the workhorses of state-of-the-art speech recognizers and machine translators that help to resolve language ambiguities by placing higher probability on more likely original underlying word strings.

However, a natural language is perhaps one of the most intriguing stochastic processes, in which messages are encoded via complex, hierarchically organized sequences. The local lexical structure of the sequence sometimes conveys surface information, while the syntactic structure, encoding long range dependencies, carries deeper semantic information.

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. 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, 2000; Zhang, 2008], and it has proven remarkably difficult to improve on n-grams.

Research groups [Brants et al., 2007; Emami et al., 2007; Zhang et al., 2006] 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 [Zhang, 2008] did not observe much improvement beyond 6-grams. Thus, there is a dire need for developing novel approaches to language modeling.

1.3 Evaluation of LMs

1.3.1 Perplexity (PPL)

One method of LM evaluation is *perplexity* (PPL) [Jelinek, 1998]. If we want to compare two models LM_1 and LM_2, they predict W_{corpus} respectively, with the probability of $P_{LM_1}(W_{corpus})$ and $P_{LM_2}(W_{corpus})$. W_{corpus} is denoted as a set of real-world English sentences, $|W_{corpus}| = N$, is its token number. We consider LM_1 to be a better model if $P_{LM_1}(W_{corpus}) > P_{LM_2}(W_{corpus})$. PPL is a direct transformation of the log probability of an English corpus.

$$PPL_{LM}(W_{corpus}) = \exp(-\frac{1}{N}\sum_{i=1}^{N}\log\left[P_{LM}(w_i|W_{i-1})\right])$$

A better LM should get a lower PPL value on a testset of real English sentences. It is often possible to achieve lower perplexity on more specialized corpora, as they are more predictable.

1.3.2 BLEU

One alternative approach to evaluate the quality of a LM is to evaluate the LM within a machine translation system. One of the most widely used metrics for assessing the translation quality is BLEU score [Papineni et al., 2002]. BLEU (Bilingual Evaluation Understudy) is a measure that evaluates the quality of text which has been machine-translated from one natural language to another. The quality is considered to be the correspondence between a machine's output and a reference. Generally speaking, the BLEU score calculates the overlapping rate of n-gram between the translation hypotheses and one of their references, on different orders of n, n=1,2,3 and 4. The formal expression of BLEU can be found in [Papineni et al., 2002].

BLEU is designed to approximate human judgment at a corpus level, and performs badly if used to evaluate the quality of individual sentences. Moreover, intelligibility or grammatical correctness are not taken into account. However, BLEU is still the most widely used evaluation metric for machine translation.

1.4 Research Motivation

As explained in Section 1.2, the traditional n-gram model ignores the long range dependency between the words. Over the past two decades, more sophisticated models have been developed that outperform n-grams, which respectively exploit sentence-level syntactic structure and document-level semantic information. These are mainly:

- *Syntactic language models* [Charniak, 2001; Chelba and Jelinek, 2000; Jelinek, 2004], which effectively exploit sentence level syntactic structure of natural language.

- *Topic language models* [Bellegarda, 2000; Gildea and Hofmann, 1999; Wallach, 2006] that exploit document level semantic content.

Unfortunately, each of these language models only targets some specific, distinct linguistic phenomena; thus, each captures and exploits different aspects of natural language regularity. A natural question we should ask is whether and/or how we can construct more complex and powerful but computationally tractable language models by integrating many existing and/or emerging language model components, with each component focusing on specific linguistic phenomena.

1.5 Our main approaches

- We develop a composite language model that integrates n-gram models, structured language model (SLM) and probabilistic latent semantic analysis (PLSA), which provides a unified probabilistic framework to seamlessly combine word-level lexical and sentence-level syntactic and document-level semantic information by directed Markov random fields.

- We generalize Jelinek and Mercer's original recursive mixing scheme to handle the smoothing of the composite LM.

- Inspired by SLM in [Chelba and Jelinek, 2000], we propose an N-best list approximate EM algorithm for training the composite LM. In addition, we propose a follow-up EM algorithm to boost the performance.

- We resort to a client-server architecture for a distributed version of the composite language models in both N-best list approximate EM and follow-up EM.

- Finally, an A^*-best lattice rescoring procedure is developed, such that the distributed composite language model is integrated into a phrase-based decoder in Moses, a state-of-the-art statistical machine translation system.

1.6 Contributions

This dissertation makes the following contributions:

- The composite language models give over 30% perplexity reduction in comparison with the standard n-gram models.
- When they are exploited in the MT systems by reranking N-best lists and rescoring the lattices from a phrase-based MT decoder, we receive a 0.7%∼0.8% increase on BLEU score and meanwhile a significant improvement on human-judged readability.
- Due to the distributed version of the composite language models, we manage to increase our training corpus over a billion tokens .
- We exploit Zangwill's global convergence theorem to prove the convergence of the N-best approximate EM for the composite language model.

As far as we know, this is the first work that builds a complex large scale distributed language model with a principled approach that simultaneously exploits syntactic, semantic and lexical regularities more powerful than n-grams trained on a very large corpus.

1.7 Thesis Layout

The thesis is organized as follows:

- In Chapter 2, we first introduce SLM and PLSA respectively. Then, we discuss some previous work about the integration of language models.
- In Chapter 3, we propose a composite language model, which integrates the lexical, syntactic and semantic information of the natural language under the direct Markov random field paradigm.

- In Chapter 4, we show how to train this composite model by an N-best list approximate EM algorithm and a follow-up EM algorithm to improve word prediction power, we proof the convergence of N-best list approximate EM algorithm. To defy 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.

- In order to handle a large scale corpus which is up to a billion tokens, we demonstrate how to implement these algorithms under a distributed computing environment and how to store and exploit this language model on a supercomputer in Chapter 5.

- In Chapter 6, we propose a lattice rescoring framework, such that the proposed composite language model is integrated into the phrase-based decoder of Moses, a state-of-the-art open-sourced machine translation system.

- In Chapter 7 and 8, we present a series of experimental results. We first discuss the effectiveness of the composite LM on PPL. Then, we exploit our proposed large scale distributed composite language model in machine translation system on N-best list reranking and lattice rescoring tasks, and examine the performance on BLEU score.

- In Chapter 9, we conclude the research and point out the directions of future work.

2

Related Work

Our research focuses on how to integrate n-gram model with more sophisticated language models, which convey various aspects of natural language from different levels and long range dependency between the words. In this chapter, we first briefly introduce SLM, an example of syntactic language models, and PLSA, an example of topic language models. Since this thesis is mainly concentrated on how to effectively combine n-gram, SLM and PLSA, we will also discuss two established techniques for LM combination.

2.1 Structured language model

SLM proposed in [Chelba and Jelinek, 1998; Chelba and Jelinek, 2000; 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 POS tags, and the internal 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 appended 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

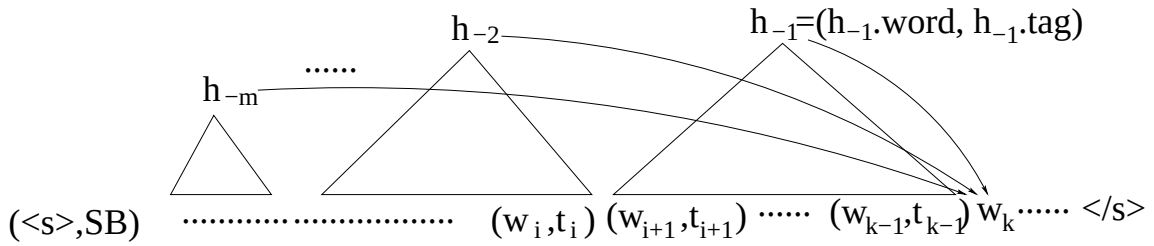

Figure 2.1: Structured language model.

from the beginning of the sentence up to the current position k and W_kT_k 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) and $\mathcal{H} = \mathcal{V} \times \mathcal{O}_{\mathrm{NT}}$ and $\mathcal{O}_{\mathrm{NT}}$ denotes the set of non-terminal label (NTlabel), or in the case of a root-only tree (word, POS tag) and $\mathcal{H} = \mathcal{V} \times \mathcal{O}$ and $\mathcal{O}$ 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 m-th 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 $p(w_{k+1}|h_{-m}^{-1})$, as shown in Figure , and then passes the 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 $p(a|h_{-m}^{-1})$;

$a \in \mathcal{A} = \{(\text{unary}, \text{NTlabel}), (\text{adjoin} - \text{left}, \text{NTlabel}), (\text{adjoin} - \text{right}, \text{NTlabel}), \text{NULL}\}$

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. The CONSTRUCTOR is repeatedly called until the CONSTRUCTOR hits NULL, the headword indexing and current parse structure stay as they are, and the CONSTRUCTOR passes control to the WORD-PREDICTOR.

SLM repeats the three steps above until all the sentences are generated. Figure 2.2 illustrates the operations of the three components above, when a word is predicted. In Figure 2.2, the part in black indicates the word-parse k-prefix when the words "the contract ended with a loss of 7" with their POS tags has been generated. The two words, "the" and "contract", form a head node whose headword is "contract_NP". So do the two words, "a" and "loss", with the headword "loss_NP". A new word "cents" (marked in red) is generated by WORD-PREDICTOR by the history of the exposed headwords. Then, TAGGER assigns the POS tag "NNS" (marked in green) to the generated word. CONSTRUCTOR runs by three steps (marked as blue) , each of which builds the partial parse by one step, until the NULL operation is generated (not shown in the figure). After that, a new word will be generated by WORD-PREDICTOR.

T_k is a hidden variable, which has to be marginalized by all the choice of T_k. SLM develops a multi-stack search for pruning the partial parses T_k. SLM predicts the next word w_{k+1} by marginalizing the WORD-PREDICTOR over all T_k survived in

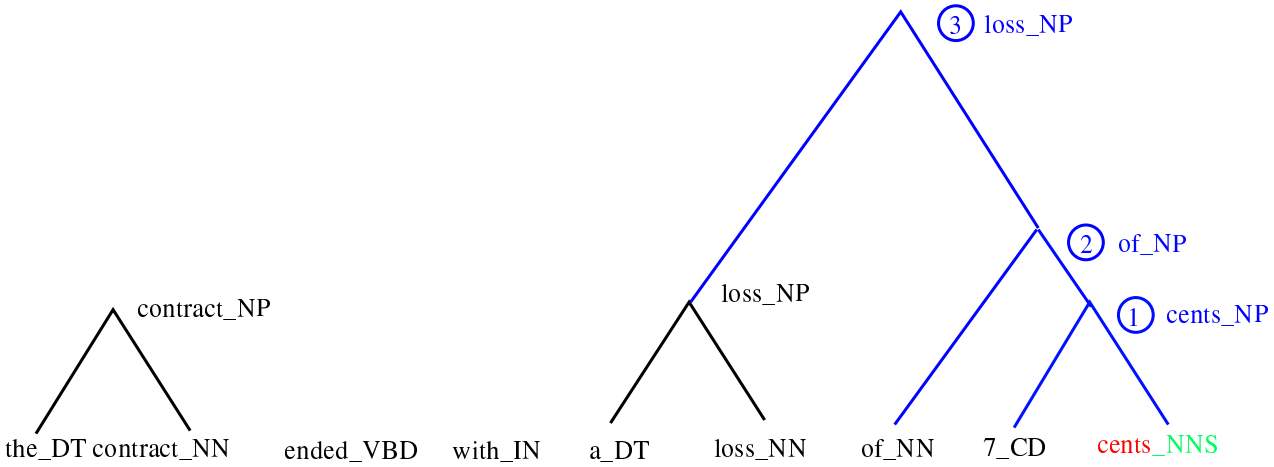

Figure 2.2: The update of a partial parse tree when a new word is predicted.

the stacks S_k of multi-stack search procedure, formulated as,

$$P(w_{k+1}|W_k) = \sum_{T_k \in S_k} P(w_{k+1}|T_k W_k) \cdot \frac{P(T_k W_k)}{\sum_{\bar{T}_k \in S_k} P(\bar{T}_k W_k)}$$

where $P(w_{k+1}|T_k W_k) = P(w_{k+1}|h_{-m}^{-1})$ for m-SLM.

Thus, SLM is essentially a generalization of a shift-reduce parser [Aho and Ullman, 1972] with *adjoin* corresponding to *reduce* and *predict* to *shift*. See detailed description about SLM in [Chelba and Jelinek, 1998; Chelba and Jelinek, 2000; Chelba, 2000; Jelinek, 2004]. As an example taken from [Jelinek, 2004], Figure 2.3 shows a complete parse where SB/SE is a distinguished POS tag for <s>/</s> respectively, (<s>,TOP) is the only allowed head, and (</s>,TOP') is the head of any constituent that dominates </s> but not <s>. In Figure 2.3, at the time just after the word "as" is generated, the exposed headwords are "<s>_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, ..."

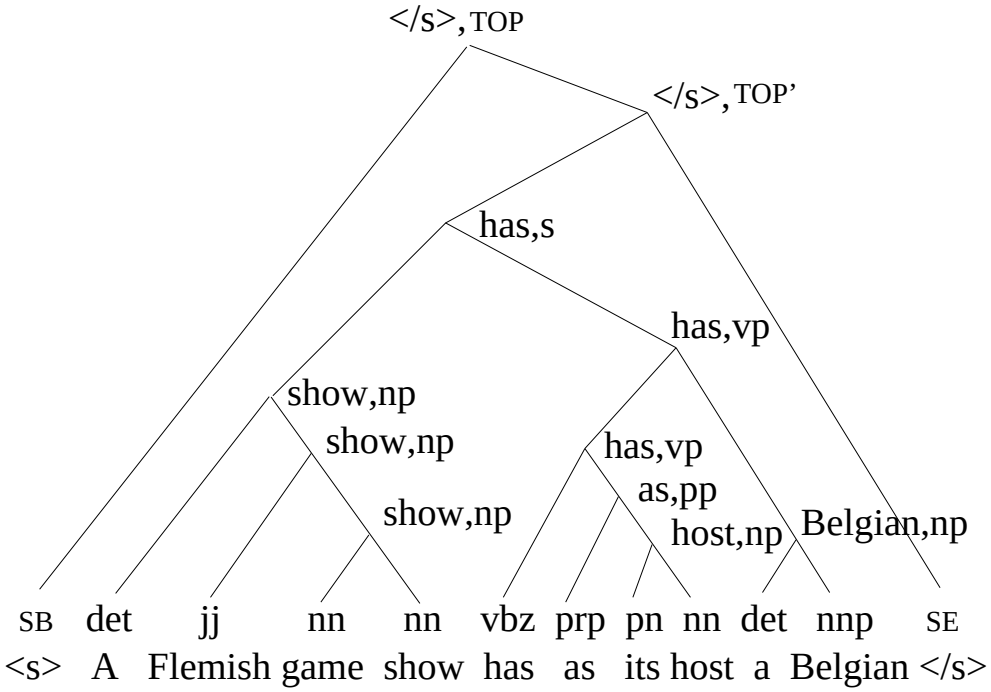

Figure 2.3: A complete parse tree by the structured language model.

2.2 Probabilistic latent semantic analysis (PLSA)

As shown in Figure 2.4, 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 $p(w|g)$.

Since only one pair of (d, w) is being observed, as a result, the joint probability model is,

$$p(d, w) = p(d) \sum_{g} p(w|g)p(g|d)$$

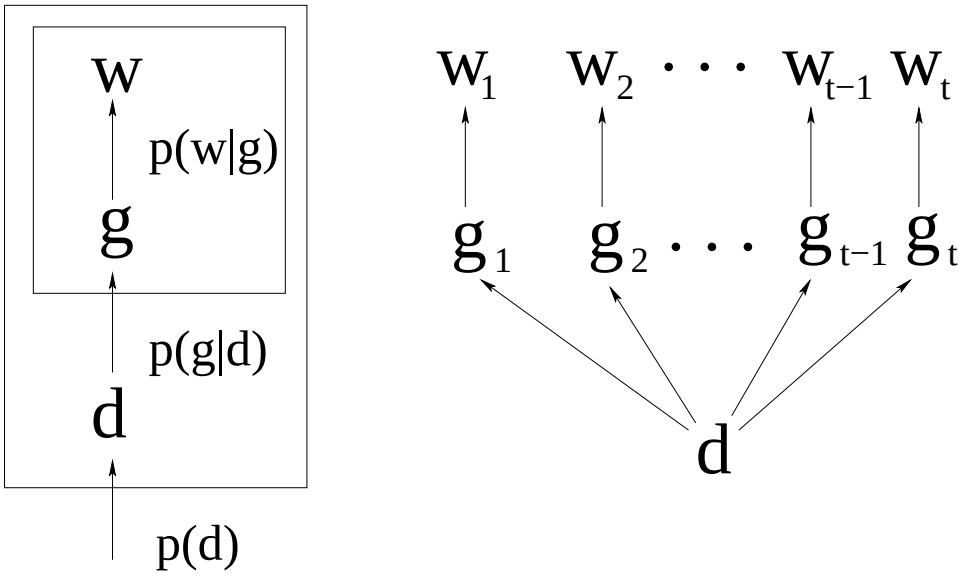

Figure 2.4: The PLSA model.

Typically for PLSA, the number of documents and vocabulary size are much larger than the size of latent semantic class variables.

2.3 Language model combination

Several techniques for combining language models have been investigated.

2.3.1 Linear Interpolation

The most commonly used method is *linear interpolation* [Jelinek and Mercer, 1980; Goodman, 2001], where each individual model is trained separately and then combined by a weighted linear combination, formulated as following,

$$P(w_k|W_k) = \sum_{i=0}^{M} \lambda_i P_{LM_i}(w_k|W_k)$$

where $\sum_{i=0}^{M} \lambda_i = 1$.

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 to capture subtleties in each of the component models.

2.3.2 Maximum entropy approach

The second method is based on *maximum entropy philosophy* which became very popular in machine leaning and natural language processing communities due to the work in [Berger et al., 1996; Pietra et al., 1997; Rosenfeld, 1996]. In fact, for a complete data case, maximum entropy is nothing but maximum likelihood estimation for undirected Markov random fields (MRFs) [Berger et al., 1996; Pietra et al., 1997].

When the idea is applied to the combination of n-gram, SLM and PLSA, the MLE can be formulated to maximize a fully-labeled sentence,

$$\max_{\underline{\lambda}} P_{\underline{\lambda}}(W,T,G) = \frac{1}{Z_{\underline{\lambda}}}\exp(<\underline{\lambda}, \#(W,T,G)>)$$

where $G = (g_1, g_2 ... g_{|L|})$ is the topic sequence for all words in W and

$$Z_{\underline{\lambda}} = \sum_{W,T,G} \exp(<\underline{\lambda}, \#(W,T,G)>$$

is one global normalization factor to ensure a proper distribution.

However, as stated in [Wang et al., 2005b], there are *two weaknesses* with maximum entropy approach. The first weakness is that this approach can only model distributions over explicitly observed features, but there are hidden information in natural language such as syntactic structure and semantic topic. The second one is that if the statistical model is too complex, it becomes intractable to estimate model parameters, computationally very expensive Markov chain Monte Carlo (MCMC) sampling methods [Mark et al., 1996; Rosenfeld, 2000; Rosenfeld et al., 2001] would have to be employed. One way to overcome the first hurdle is to use a preprocessing

tool to extract hidden features, for example, Rosenfeld [Rosenfeld, 1996] used mutual information clustering method to find word pair triggers, then combine these triggers with trigrams through maximum conditional entropy approach to allow the discourse topic to influence word prediction; Khudanpur and Wu [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 again combine these features with trigrams through maximum conditional entropy approach to form a syntactic, semantic and lexical language model. Their work is a special case of exponential language models [Chen, 2009].

Wang et al. [Wang et al., 2005a; Wang et al., 2012] have proposed the *latent maximum entropy (LME)* principle, which extends standard maximum entropy estimation by incorporating hidden dependency structure, but still the LME would not overcome the second hurdle.

2.3.3 Directed Markov random fields

The third method is the *directed Markov random fields* [Wang et al., 2005b] that overcomes the two weaknesses in maximum entropy approach. Wang et al. used this approach to combine trigram, probabilistic context free grammar (PCFG) and probabilistic latent semantic analysis (PLSA), 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 of the effect of n-gram and PLSA while remaining the same cubic time complexity. When applying to the Wall Street Journal corpus with 40 million tokens, they achieved moderate perplexity reduction.

Since the probabilistic dependency structure in structured language model (SLM) [Chelba and Jelinek, 2000; Chelba, 2000] is more complex and powerful than that in a PCFG, Wang et al. [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 composite n-gram, SLM and PLSA language model from a general EM [Dempster et al., 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 6th order of sentence length time complexity. Unfortunately, there are no experimental results reported.

In this dissertation, we study the same composite n-gram, SLM, and PLSA under the directed MRF framework as in [Wang et al., 2006]. The composite n-gram/SLM/PLSA language model under the directed Markov random field paradigm is first introduced. Instead of using the 6th order generalized inside-outside algorithm proposed in [Wang et al., 2006], we show how to train this composite model by an N-best list approximate EM algorithm that has linear time complexity and a follow-up EM algorithm to improve word prediction power, we proof the convergence of N-best list approximate EM algorithm. To defy 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. Then we describe how to use the model for testing. Since 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 experiment section, 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 reranking the N-best list from Hiero [Chiang,

2005; Chiang, 2007], a state-of-the-art parsing-based machine translation system, we achieve significantly better translation quality measured by the BLEU score. Also, we propose an A^*search algorithm performed on lattices generated by a state-of-the-art phase-based MT system, in order to integrate the composite LM into MT system in a more sophisticated approach.

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]. 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].

3

A Composite Language Model

In this chapter, we introduce the construction of the composite language model, which combines n-gram, SLM and PLSA under directed Markov random fields (MRF). We first introduce the concept of directed MRFs and its difference from undirected MRFs. Then, we present the structure of the composite language model. Finally, we generalize Jelinek and Mercer's original recursive mixing scheme for smoothing to handle the situation where the context is a mixture of Markov chains in this composite language model.

3.1 Directed Markov random field

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,

$$p(x) = \prod_{\tau\in\Gamma} \kappa^\tau(x_\tau, x_{pa(\tau)}) \tag{3.1}$$

Here $pa(\tau)$ denotes the set of parent states of τ. If the recursive factorization respects to a graph, then we have a Bayesian network [Lauritzen, 1996]. But broadly speaking, the recursive factorization can respect to a more complicated representation other

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 not be described as a Bayesian network [McAllester et al., 2004].

3.1.1 Undirected MRFs vs directed MRFs

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 intersection between directed MRFs and undirected MRFs are decomposable MRFs [Lauritzen, 1996]. See Figure 3.1 for an illustration, where each model can be represented by either a directed MRF or an undirected MRF, there is a one-one mapping for both representation. One example is the n-gram model which can be represented by either a Markov chain or a maximum entropy model. To train a Markov chain is simple, which is nothing but relative frequency estimate with proper smoothing, whereas to train an n-gram maximum entropy model is quite expensive [Jelinek, 1998] since the iterative training algorithm needs feature expectation and normalization in the inner loop plus optimization such as iterative scaling, coordinate descent, quasi-Newton etc. in the outer loop. PCFG is another example of decomposable MRFs as proven by Chi [Chi, 1999].

3.2 Model structure

When combining n-gram, m order SLM, and PLSA together to build a composite generative language model under the directed MRF paradigm [Wang et al., 2005b; Wang et al., 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; however, the WORD-PREDICTORs in n-gram, m-SLM, and PLSA are combined to form a stronger WORD-PREDICTOR that gen-

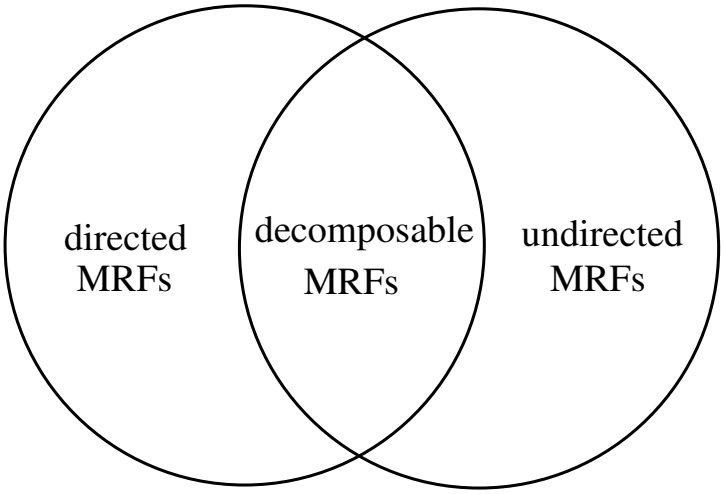

Figure 3.1: Undirected MRFs versus directed MRFs, the intersection are decomposable MRFs, n-gram, PCFG and SLM are examples of decomposable MRFs, and maximum entropy models correspond to undirected MRFs with complete data.

erates 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 3.2 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 normalization constraints for the parameters of each model component, WORD-

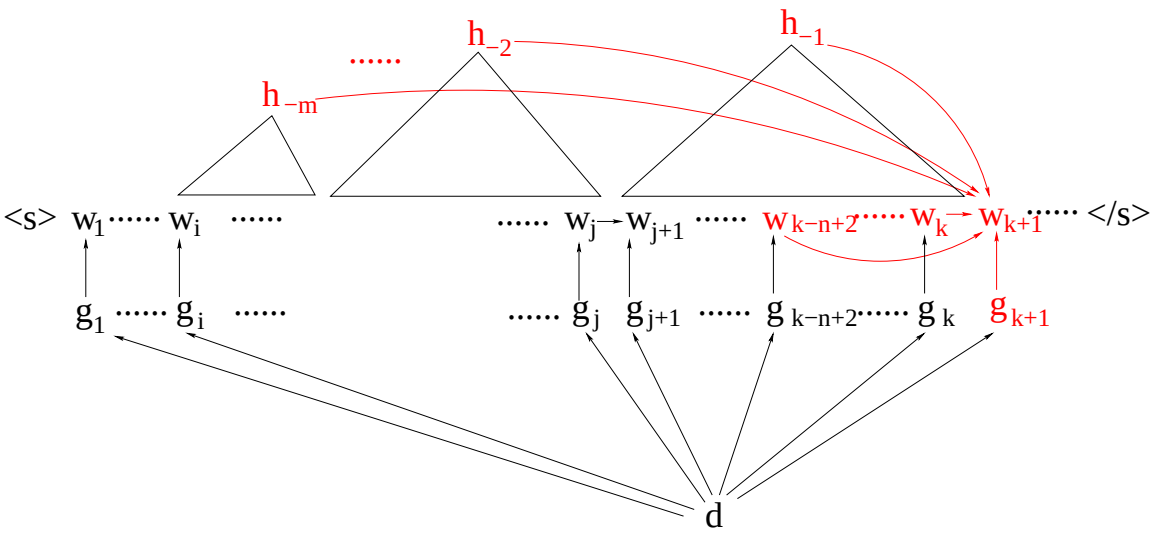

Figure 3.2: A composite n-gram/m-SLM/PLSA language model where the hidden information is the parse tree T and semantic content g. n-gram encodes local word interactions, m-SLM models sentence's syntactic structure and PLSA captures document's semantic content, all interact together to constrain the generation 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.

PREDICTOR, TAGGER, CONSTRUCTOR, SEMANTIZER, i.e.,

$$\sum_{w \in \mathcal{V}} p(w|w_{-n+1}^{-1} h_{-m}^{-1} g) = 1 \tag{3.2}$$

$$\sum_{t \in \mathcal{O}} p(t|w h_{-m}^{-1}.\text{tag}) = 1 \tag{3.3}$$

$$\sum_{a \in \mathcal{A}} p(a|h_{-m}^{-1}) = 1 \tag{3.4}$$

$$\sum_{g \in \mathcal{G}} p(g|d) = 1 \tag{3.5}$$

If we look at the example in Figure 2.3, 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 next word, such as "a", based on m most recently exposed headwords "<s>-SB, show-np, has-vp," but for the composite model, the WORD-PREDICTOR predicts next word "a" based on m most recently exposed headwords "<s>-SB, show-np, has-vp," n-grams "as its host" and a topic g. These are the only difference between SLM and our proposed composite language model.

3.3 Smoothing Techniques

By maximum likelihood estimation (MLE) [Jurafsky and Martin, 2008], the probability of the predicted word w_k given its history is

$$p_{MLE}(w_k|w_{k-n+1}^{k-1}) = \frac{C(w_{k-n+1}^{k})}{C(w_{k-n+1}^{k-1})}$$

where $C(w_{k-n+1}^{k})$ is the number of the occurrence of w_{k-n+1}^{k} in the corpus. MLE produces poor estimation when one of the two counts is either zero or non-zero but still small. Widely-used techniques for smoothing include Good Turning discounts, linear interpolation, Katz backoff and Kneser-Ney smoothing [Jurafsky and Martin, 2008].

3.3.1 Recursive mixing scheme for smoothing

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, 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; Chelba and Jelinek, 2000; 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 original recursive mixing scheme [Jelinek and Mercer, 1980] to handle the situation where the context is a mixture of Markov chains. Factored language (FL) model [Bilmes and Kirchhoff, 2003] is close to the smoothing technique we proposed 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, while our approach strictly respects the order of Markov chains for word sequence and headword sequence, since we believe natural language tightly follows these orders; Moreover, FL uses backoff technique, we use linear interpolation.

Consider a composite trigram/2-SLM/PLSA language model, Figure 3.3 illustrates a lattice formed all possible conditional probabilistic models and relative frequency estimates of different orders along each of 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 interpola-

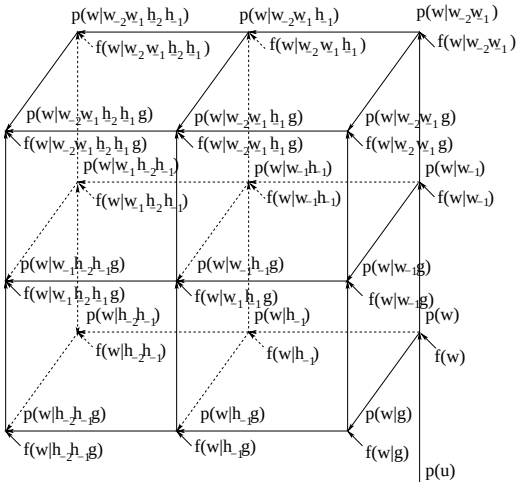

Figure 3.3: Recursive linear interpolation lattice to estimate WORD-PREDICTOR $p(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$ of the composite trigram/2-SLM/PLSA language model, where $\mathcal{U}$ is the vocabulary in which the predicted random variable w takes values and $p(\mathcal{U})$ denotes uniform distribution 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.

tion 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 its relative frequency estimate $f(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$,

$$\begin{aligned} p(w|w_{-2}w_{-1}h_{-2}h_{-1}g) &= \lambda_w(w_{-2}w_{-1}h_{-2}h_{-1}g) \cdot p(w|w_{-1}h_{-2}h_{-1}g) \\ &\quad +\lambda_h(w_{-2}w_{-1}h_{-2}h_{-1}g) \cdot p(w|w_{-2}w_{-1}h_{-1}g) \\ &\quad +\lambda_g(w_{-2}w_{-1}h_{-2}h_{-1}g) \cdot p(w|w_{-2}w_{-1}h_{-2}h_{-1}) \\ &\quad +(1 - \lambda_w(w_{-2}w_{-1}h_{-2}h_{-1}g) - \lambda_h(w_{-2}w_{-1}h_{-2}h_{-1}g) \\ &\quad -\lambda_g(w_{-2}w_{-1}h_{-2}h_{-1}g)) \cdot f(w|w_{-2}w_{-1}h_{-2}h_{-1}g) \end{aligned} \tag{3.6}$$

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, $f(w|w_{-2}w_{-1}h_{-2}h_{-1}g) = \frac{C(w_{-2}w_{-1}wh_{-2}h_{-1}g)}{C(w_{-2}w_{-1}h_{-2}h_{-1}g)}$, $C(w_{-2}w_{-1}wh_{-2}h_{-1}g)$ 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 EM algorithm [Dempster et al., 1977].

4

Training and Testing Algorithms

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

$$\hat{\mathcal{L}}(\mathcal{D}, p) = \prod_{d \in \mathcal{D}} \left(\left(\prod_{l} \left(\sum_{G^l} \left(\sum_{T^l} P_p(W^l, T^l, G^l | d) \right) \right) \right) p(d) \right) \tag{4.1}$$

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,

$$\begin{aligned} P_p(W^l, T^l, G^l | d) \;=\; & \prod_{g \in \mathcal{G}} \Big(p(g|d)^{\#(g, W^l, G^l, d)} \\ & \left(\prod_{w, w_{-1}, \cdots, w_{-n+1} \in \mathcal{V}} p(w | w_{-n+1}^{-1} h_{-m}^{-1} g)^{\#(w_{-n+1}^{-1} w h_{-m}^{-1} g, W^l, T^l, G^l, d)} \right. \\ & \left. \left. \prod_{t \in \mathcal{O}} p(t | w h_{-m}^{-1}.\text{tag})^{\#(t, w h_{-m}^{-1}.\text{tag}, W^l, T^l, d)} \prod_{a \in \mathcal{A}} p(a | h_{-m}^{-1})^{\#(a, h_{-m}^{-1}, W^l, T^l, d)} \right) \right) \end{aligned} \tag{4.2}$$

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, $\#(twh_{-m}^{-1}.\text{tag}, W^l, T^l, d)$ is the count of tag t predicted by word w and the tags of m most recently 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.

Let

$$\mathcal{L}(\mathcal{D}, p) = \prod_{d \in \mathcal{D}} \left(\prod_{l} \left(\sum_{G^l} \left(\sum_{T^l} P_p(W^l, T^l, G^l | d) \right) \right) \right) \tag{4.3}$$

then

$$\hat{\mathcal{L}}(\mathcal{D}, p) = \mathcal{L}(\mathcal{D}, p) \prod_{d \in \mathcal{D}} (p(d)) \tag{4.4}$$

Clearly when maximizing $\hat{\mathcal{L}}(\mathcal{D}, p)$ in Eq. 4.1, $p(d)$ 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 n-gram language model, we will generally ignore this term and concentrate on optimizing Eq. 4.3 in the subsequent development.

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, the work in [Wang et al., 2006] have derived a generalized inside-outside algorithm by applying the standard EM algorithm and considering the auxiliary function,

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

However, the complexity of this algorithm is the 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].

4.1 N-best list approximate EM

4.1.1 General framework

Similar to SLM [Chelba and Jelinek, 1998; Chelba and Jelinek, 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,

$$\max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, p, \mathcal{T'}_N) = \prod_{d \in \mathcal{D}} \left(\prod_l \left(\max_{\mathcal{T'}^l_N \in \mathcal{T'}_N} \left(\sum_{G^l} \left(\sum_{T^l \in \mathcal{T'}^l_N, ||\mathcal{T'}^l_N|| = N} P_p(W^l, T^l, G^l|d) \right) \right) \right) \right) \quad (4.5)$$

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 $\mathcal{D}$,

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

   That is,

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

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

- 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.

4.1.2 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 is kept due to the consideration in both computational time and resource demand, otherwise we have to use much more machines to store WORD-PREDICTOR's parameters.

We can either find both the N-best parses for each sentence and τ-best topics for each document simultaneously or separately. However, the later one is much preferred, since the first case is much more computationally expensive.

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

To extract the N-best parse trees, we adopt a synchronous, multi-stack search strategy (Figure 4.1) that is similar to the one in [Chelba and Jelinek, 1998; Chelba and Jelinek, 2000; 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

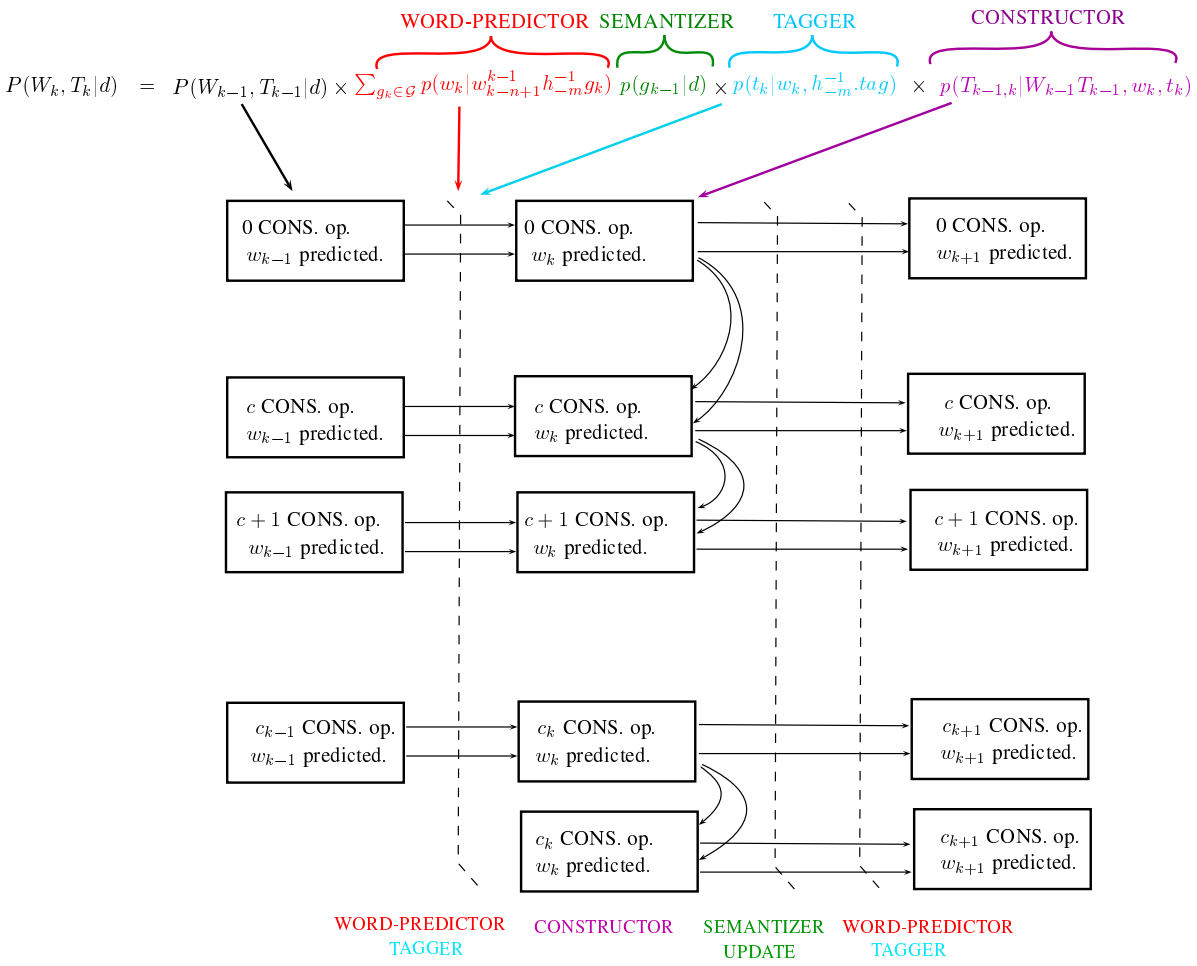

Figure 4.1: Multi-stack beam search

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 and 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. 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,

$$\begin{aligned} P_p(W_k, T_k|d) &= P_p(W_{k-1}, T_{k-1}|d) \left(\sum_{g_k \in \mathcal{G}_d} p(w_k|w_{k-n+1}^{k-1} h_{-m}^{-1} g_k) \frac{p(g_k|d)}{\sum_{g_i \in \mathcal{G}_d} p(g_i|d)} \right) \\ & \quad p(t_k|w_k, h_{-m}^{-1}.\text{tag}) p(T_{k-1,k}|W_{k-1}T_{k-1}, w_k, t_k) \end{aligned} \quad (4.6)$$

where $W_{k-1}T_{k-1}$ is the word-parse $(k-1)$-prefix; w_k is the k-th 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}, it is the parse structure built on top of T_{k-1} and the newly predicted word w_k, the $||$ notation stands for concatenation; and 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. Since 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 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.

As shown in Figure 4.1, the synchronous, multi-stack search strategy is a greedy best first search algorithm, one of the local heuristic search procedures, which does not use future cost estimate 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]. However, in practice we find that the N-best list approximate EM algorithm does converge within several iterations.

4.1.3 EM Update

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

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

$$p'(w|w_{-n+1}^{-1}h_{-m}^{-1}g) \propto \sum_{d\in\mathcal{D}}\sum_{l}\sum_{G^l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l, G^l|W^l, d)\#(w_{-n+1}^{-1}wh_{-m}^{-1}g, W^l, T^l, G^l, d) \quad (4.7)$$

$$p'(t|wh_{-m}^{-1}.\text{tag}) \propto \sum_{d\in\mathcal{D}}\sum_{l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l|W^l, d)\#(twh_{-m}^{-1}.\text{tag}, W^l, T^l, d) \quad (4.8)$$

$$p'(a|h_{-m}^{-1})) \propto \sum_{d\in\mathcal{D}}\sum_{l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l|W^l, d)\#(ah_{-m}^{-1}, W^l, T^l, d) \quad (4.9)$$

$$p'(g|d) \propto \sum_{d\in\mathcal{D}}\sum_{l}\sum_{G^l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l, G^l|W^l, d)\#(g, W^l, G^l, d) \quad (4.10)$$

In E-step, we use Eqs (4.7-4.10) 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 exponential with sentence length, we use Jelinek-style recursive formulas in the generalized inside-out-side algorithm [Jelinek, 2004] to handle the tree structure and describe the weighted forest of possible derivations [Wang et al., 2006]. However in the N-best list case considered in this paper, we just enumerate each parse tree in N-best list and compute the expected posterior count for each parse tree. For the WORD-PREDICTOR and the SEMANTIZER, we use Eq. 4.7 and Eq. 4.10, 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

$$\begin{aligned}\alpha_{k+1}^l(g|d) &= \sum_{G_k^l} P_p(W_k^l, T_k^l, w_{k-n+2}^k w_{k+1} h_{-m}^{-1} g, G_k^l|d) \\ &= P_p(W_k^l, T_k^l, w_{k-n+2}^k w_{k+1} h_{-m}^{-1} g|d) \\ &= P_p(W_k^l, T_k^l|d) p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1} g, d) \frac{p(g_{k+1}|d)}{\sum_{g_i \in \mathcal{G}_d} p(g_i|d)}\end{aligned} \tag{4.11}$$

where W_k^l is the word k-prefix for sentence W^l, 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 Eq. 4.6 as

$$\begin{aligned}\alpha_{k+1}^l(g|d) &= \left(\sum_{g_k \in \mathcal{G}_d} \alpha_k^l(g_k|d) \right) p(t_k|w_k, h_{-m}^{-1}.\text{tag}) p(T_{k-1,k}^l|W_{k-1}^l T_{k-1}^l, w_k, t_k) \\ &\quad p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1} g, d) \frac{p(g_{k+1}, d)}{\sum_{g_i \in \mathcal{G}_d} p(g_i|d)}\end{aligned} \tag{4.12}$$

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

$$\beta_{k+1}^l(g|d) = \sum_{G_{k+1,\cdot}^l} P_p(W_{k+1,\cdot}^l, T_{k+1,\cdot}^l, G_{k+1,\cdot}^l|w_{k-n+2}^k w_{k+1} h_{-m}^{-1} g, d) \tag{4.13}$$

where $W_{k+1,\cdot}^l = w_{k+2}^l, \cdots, < /s >$ 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_{k+1}^l that generates parse tree T^l, $T^l = T_{k+1}^l || T_{k+1,\cdot}^l$, $G_{k+1,\cdot}^l = g_{k+2}, \cdots$, is the semantic subsequence in G^l relevant to $W_{k+1,\cdot}^l$. Again it is easy to see that the backward vector $\beta^l(g|d)$ can be recursively computed in a backward manner as

$$\begin{aligned}\beta_{k+1}^l(g|d) &= p(t_{k+1}|w_{k+1}, h_{-m}^{-1}.\text{tag}) p(T_{k,k+1}^l|W_k^l T_k^l, w_{k+1}, t_{k+1}) \\ &\quad \sum_{g_{k+2} \in \mathcal{G}_d} p(w_{k+2}|w_{k-n+3}^k h_{-m}^{-1} g_{k+2}, d) \frac{p(g_{k+2}|d)}{\sum_{g_i \in \mathcal{G}_d} p(g_i|d)} \\ &\quad \sum_{G_{k+2,\cdot}^l} P_p(W_{k+2,\cdot}^l, T_{k+2,\cdot}^l, G_{k+2,\cdot}^l|w_{k-n+3}^{k+1} w_{k+2} h_{-m}^{-1} g_{k+2}, d) \\ &= p(t_{k+1}|w_{k+1}, h_{-m}^{-1}.\text{tag}) p(T_{k,k+1}^l|W_k^l T_k^l, w_{k+1}, t_{k+1}) \\ &\quad \sum_{g_{k+2} \in \mathcal{G}_d} p(w_{k+2}|w_{k-n+3}^k h_{-m}^{-1} g_{k+2}, d) \frac{p(g_{k+2}|d)}{\sum_{g_i \in \mathcal{G}_d} p(g_i|d)} \beta_{k+2}^l(g_{k+2}|d)\end{aligned} \tag{4.14}$$

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

$$\begin{aligned}
&\sum_{G^l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l,G^l|W^l,d)\#(w_{-n+1}^{-1}wh_{-m}^{-1}g,W^l,T^l,G^l,d) \qquad (4.15)\\
=&\sum_{G^l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l,G^l,W^l|d)\#(w_{-n+1}^{-1}wh_{-m}^{-1}g,W^l,T^l,G^l,d)/P_p(W^l|d)\\
=&\sum_l\sum_k \alpha_{k+1}^l(g|d)\beta_{k+1}^l(g|d)\delta(w_{k-n+2}^k w_{k+1}h_{-m}^{-1}g_{k+1}=w_{-n+1}^{-1}wh_{-m}^{-1}g)/P_p(W^l|d)
\end{aligned}$$

where $P_p(W^l|d)=\sum_{G^l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N}P_p(T^l,G^l,W^l|d)=\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N}P_p(T^l,W^l|d)$ and $P_p(T^l,W^l|d)$ is recursively computed by Eq. 4.6 through traversing the l-th parse tree $T^l\in\mathcal{T}_N^l$ of sentence W^l from left-to-right, $\delta(\cdot)$ is an indicator function. The expected count of g for the SEMANTIZER on sentence W^l in document d is

$$\begin{aligned}
&\sum_{G^l}\sum_{T^l\in\mathcal{T}_N^l\in\mathcal{T}_N} P_p(T^l,G^l|W^l,d)\#(g,W^l,G^l,d) \qquad (4.16)\\
=&\sum_l\sum_k \alpha_{k+1}^l(g|d)\beta_{k+1}^l(g|d)p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1}g)/P_p(W^l|d)
\end{aligned}$$

For the TAGGER and the CONSTRUCTOR, we use Eq. 4.8 and Eq. 4.9, and the expected count of each event of twh_{-m}^{-1}.tag and ah_{-m}^{-1} 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 $P_p(T^l|W^l,d)=P_p(T^l,W^l|d)/\sum_{T^l\in\mathcal{T}^l}P_p(T^l,W^l|d)$, i.e., $P_p(T^l|W^l,d)\#(twh_{-m}^{-1}.\text{tag},W^l,T^l,d)$ and $P_p(T^l|W^l,d)\#(ah_{-m}^{-1},W^l,T^l,d)$ 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 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; Chelba and Jelinek, 2000; Chelba, 2000].

In 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 in Section 3.3.1 is then used to obtain a smooth probability estimate for each model component.

4.1.4 Proof of convergence

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 $\theta_i \rightarrow \theta, \theta_i \in \Theta$ and $\lambda_i \rightarrow \lambda, \lambda_i \in \mathcal{M}(\theta_i)$ implies $\lambda \in \mathcal{M}(\theta)$. For point-to-point map, continuity implies closeness. Then the global convergence theorem [Zangwill, 1969] states as

Theorem 1. *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 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 [Wu, 1983] to prove the convergence of standard EM algorithm [Dempster et al., 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. However, we encounter one difficulty at this point due to the maximization operator in Eq. 4.5, 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. Since the composite model is essentially a mixture model of curved exponential family through a complex hierarchy, there is a closed form solution for $\tilde{Q}(p', p, \mathcal{T}_N)$ function irrespective to N-best

list parse trees, so the N-best list approximate EM algorithm is a one-to-one map. Since $\tilde{Q}(p', p, \mathcal{T}_N)$ is continuous in both p' and p, the map is closed, thus condition (i) is satisfied.

To check the 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:

$$\check{\mathcal{T}}_N = \arg\max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, \check{p}, \mathcal{T'}_N) \tag{4.17}$$

$$\bar{\mathcal{T}}_N = \arg\max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, \bar{p}, \mathcal{T'}_N) \tag{4.18}$$

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

$$\bar{p} = \arg\max_{p'} \tilde{Q}(p', \check{p}, \check{\mathcal{T}}_N) \tag{4.19}$$

Then,

$$\max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, \bar{p}, \mathcal{T'}_N) \geq \tilde{\mathcal{L}}(\mathcal{D}, \bar{p}, \check{\mathcal{T}}_N) \tag{4.20}$$

$$\geq \tilde{\mathcal{L}}(\mathcal{D}, \check{p}, \check{\mathcal{T}}_N) \tag{4.21}$$

$$\geq \max_{\mathcal{T'}_N} \mathcal{L}(\mathcal{D}, \check{p}, \mathcal{T'}_N) \tag{4.22}$$

The inequality in Eq. 4.20 is strict unless $\check{\mathcal{T}}_N = \bar{\mathcal{T}}_N$ which results in $\bar{p} \in \mathcal{M}(\bar{p})$. Using results proven in [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 Eq. 4.21 is strict. Finally, the inequality in Eq. 4.22 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.

4.2 Follow-up EM

As explained in [Chelba and Jelinek, 2000; 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 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 available. From the causal relationship among the parameters of the composite n-gram/m-SLM/PLSA, we have:

$$\begin{aligned} P_p(w_{k+1}|W_k,d) &= \sum_{T_k\in Z_k, g_{k+1}\in\mathcal{G}_d} P_p(w_{k+1},T_k,g_{k+1}|W_k,d) \\ &= \sum_{T_k\in Z_k, g_{k+1}\in\mathcal{G}_d} P_p(w_{k+1}|W_k,T_k,g_{k+1},d)P_p(T_k|W_k,d)\frac{p(g_{k+1}|d)}{\sum_{g_i\in\mathcal{G}_d}p(g_i|d)} \\ &= \sum_{h_{-m}^{-1}\in T_k; T_k\in Z_k, g_{k+1}\in\mathcal{G}_d} p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1} g_{k+1})P_p(T_k|W_k,d)\frac{p(g_{k+1}|d)}{\sum_{g_i\in\mathcal{G}_d}p(g_i|d)} \end{aligned} \tag{4.23}$$

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, 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 $W_k = w_0,\cdots,w_k$, $G_k = g_1,\cdots,g_k, g_i\in\mathcal{G}_d, i=1,\cdots,k$ 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

$$\tilde{\mathcal{L}}(\mathcal{D}, p) = \prod_{d \in \mathcal{D}} \prod_{l} \left(P_p(W^l|d) \right) \tag{4.24}$$

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 Eq. 4.3, we ignore the ancillary term $p(d)$ in Eq. 4.24.

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 maximizing Eq. 4.24 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 (HMM) 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 $P_p(T_k, g_{k+1}|w_{k+1}, W_k, d)$, i.e.,

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

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

4.3 Use the model for testing

When a language model is used in one-pass decoders of speech recognition and phrased-based machine translation 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 word level given by Eq. 4.23 one at a time. Since 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 "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, i.e., a pseudo-document $\tilde{d}_k = (W_k, S)$, here S is the set of previous sentences of a document in test data, while holding the other parameters fixed. Wang et al. [Wang et al., 2005b] use online gradient ascent to re-estimate these parameters. We use three methods to re-estimate these parameters:

- *one step online EM*:

$$p(g|\tilde{d}_k) = \gamma \frac{\sum_{h_{-m}^{-1} \in T_{k-1}; T_{k-1} \in Z_{k-1}} p(w_k|w_{k-n+1}^{k-1} h_{-m}^{-1} g) p(g|\tilde{d}_{k-1}) P_p(T_{k-1}|W_{k-1}, \tilde{d}_{k-1})}{\sum_{g \in \mathcal{G}_d} \sum_{h_{-m}^{-1} \in T_{k-1}; T_{k-1} \in Z_{k-1}} p(w_k|w_{k-n+1}^{k-1} h_{-m}^{-1} g) p(g|\tilde{d}_{k-1}) P_p(T_{k-1}|W_{k-1}, \tilde{d}_{k-1})} + (1-\gamma) p(g|\tilde{d}_{k-1}) \quad (4.25)$$

  where γ is set to be equal to $\frac{1}{|\tilde{d}_k|+1}$.

- *online EM with fixed learning rate* : $\gamma = 0.2$.

- *batch EM*: The standard EM algorithm where we keep the iterative procedure until convergence.

The initial values are set to be

$$p(g|\tilde{d}_0) = \frac{\sum_{d \in \mathcal{D}} \#(d) \frac{p(g|d)}{\sum_{g_i \in \mathcal{G}_d} p(g_i|d)}}{|\mathcal{D}|}$$

where for the topics that are purged, we just plug-in 0 for $p(g|d)$. $\#(d)$ is the number of words in document $d, d \in \mathcal{D}$, $|\mathcal{D}| = \sum_d \#(d)$ denotes the size of training corpus which is the total number of words in the entire training corpus.

When we use Eq. 4.23 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 7.8 gives perplexity results

which uses these three methods to re-estimate the parameters of the SEMANTIZER, where the online EM with fixed learning rate not only has the cheapest computational cost but also leads to highest perplexity reductions.

5

System Architecture

In chapter 3 and 4, we ready give a detailed explanation about the structure of the proposed composite language model, as well as the N-best list approximate EM and the follow-up EM to train the LM. In this chapter, we propose a distributed framework for the composite language model, such that the training corpus can be scaled up to 1 billion tokens. The EM algorithms in the previous chapter are performed within the cloud computing environment.

In section 5.1, we first briefly retrospect some previous work on building a large-scale distributed n-gram models, and show the difference of our proposed framework.

5.1 Related works on large scale n-gram models

When using very large corpora to train our composite language model, both the data and the parameters cannot 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 existing works use distributed architectures that follow the client-server paradigm, the real implementations are in fact different. In the work of [Zhang et al., 2006; Emami et al., 2007], the training corpora are stored 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. [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 via hashing 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.

Although MapReduce is proved to be an efficient and reliable architecture for the construction of the n-gram language models, this mechanism cannot be directly adopted into our framework for the reason that MapReduce is executed by a batch job scheduler, in which the data flow is always from mappers to reducers, and mappers and reducers cannot communicate with each other in an interactive and asynchronous pattern.

Specifically, it is not suitable if we adopt the MapReduce structure because of the following two reasons:

- The N-best list approximate EM and the follow-up EM algorithms in Chapter 4 repeat their E-step and M-step for multiple iterations. For each iterations, the model parameters of the old iteration (stored in reducers) will be inquired for the calculation of the new iteration (processed in mappers).

- Since the headwords h and the topics g are hidden variables and cannot be predicted before the E-step, such inquiries for the parameters of the old iteration have to be created asynchronously among the clients. That is, the mappers generate different patterns of $(w_k, w_{k-n+1}^{k-1} h_{-m}^{-1} g)$ during the E-step, and cannot obtain the types of $(w_k w_{k-n+1}^{k-1} h_{-m}^{-1} g)$ ahead.

Such two characteristics prevent the state-of-the-art MapReduce software from being appropriate for our problems. Consequently, we resort to a client-server paradigm for our distributed language model implemented by message passing interface (MPI).

5.2 Distributed training strategy for the composite LM

We adopt a client-server approach for training the large-scale corpus and make it suitable to perform iterations of N-best list approximate EM algorithm, see Figure 5.1.

Initialization of the model parameters is divided into two steps. (1) 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 $w_{-n+1}^{-1}wh_{-m}^{-1}g$ (WORD-PREDICTOR), twh_{-m}^{-1}.tag (TAGGER) and ah_{-m}^{-1} (CONSTRUCTOR), and finish the model initialization part on the client side. (2) 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, 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. This is the initialization of the N-best list approximate EM step. Step (2) can be implemented by the standard MapReduce.

For N-best list approximate EM, 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.

It is also divided into two steps. (1) The expected count for a particular parameter of $w_{-n+1}^{-1}wh_{-m}^{-1}g$, twh_{-m}^{-1}.tag and ah_{-m}^{-1} for the new iteration at the clients are computed, while the old parameters are needed on the server side. (2) The expected count of $w_{-n+1}^{-1}wh_{-m}^{-1}g$ are summed up and stored in one of the servers by hashing through word w_{-1}, headword h_{-1}.tag 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. The SEMANTIZER has document-specific parameters, thus the EM iterative updates are performed at each of local clients. Step (2) is also a standard MapReduce procedure, which we refer as *Model parameter gatherer*. We repeat this procedure until

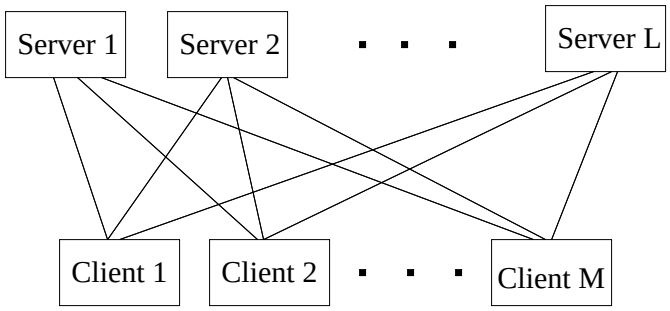

Figure 5.1: Distributed architecture is essentially a client-server paradigm: clients store partitioned data and perform E-step: compute expected counts on the client side; servers store parameters (counts) for M-step where counts of $w_{-n+1}^{-1}wh_{-m}^{-1}g$ are hashed by the last two elements in the LM 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.

convergence. Figure 5.2 illustrates the iterative training procedure of N-best list approximate EM and the follow-up EM.

We implement the client server paradigm by C++ and message passing interface (MPI) and train the composite language model on Glenn cluster of Ohio supercomputer center. Our language model software is built on MVAPICH2, an implementation of the Message Passing Interface (MPI), which is based on MPICH and optimized for the high-speed Infiniband interconnects.

Similarly, we use a distributed architecture as in Figure 5.1 to perform the follow-up EM algorithm to re-estimate WORD-PREDICTOR.

5.3 Systematic workflow

In this section, we give a high-level picture of the systematic workflow of how to train the composite language model, as shown in Figure 5.3.

1. Before the execution, a set of parameters are manually set up, mainly including the number of clients N_c, the number of servers N_s , the topic size V_g, the N-best

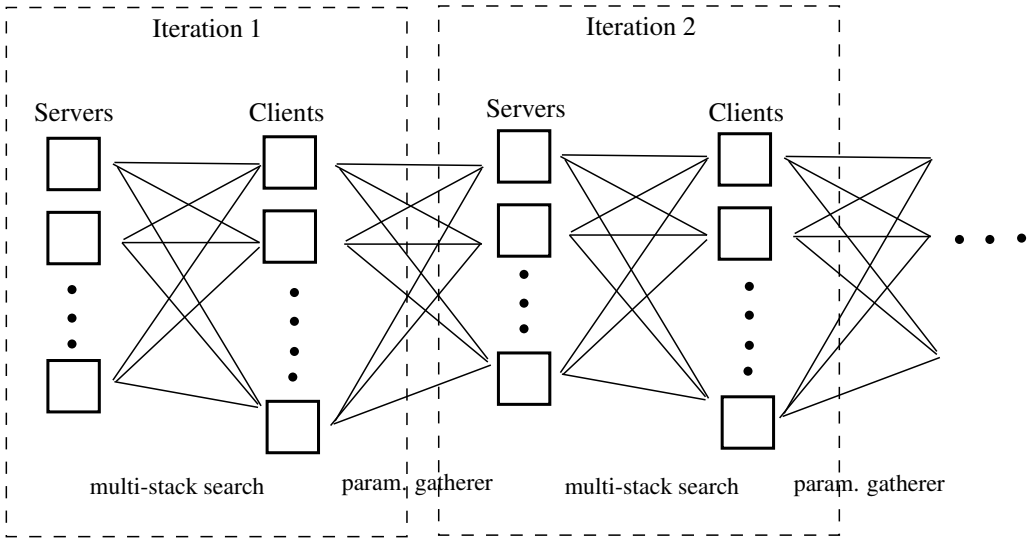

Figure 5.2: The interactive training framework of N-best list approximate EM and the follow-up EM.

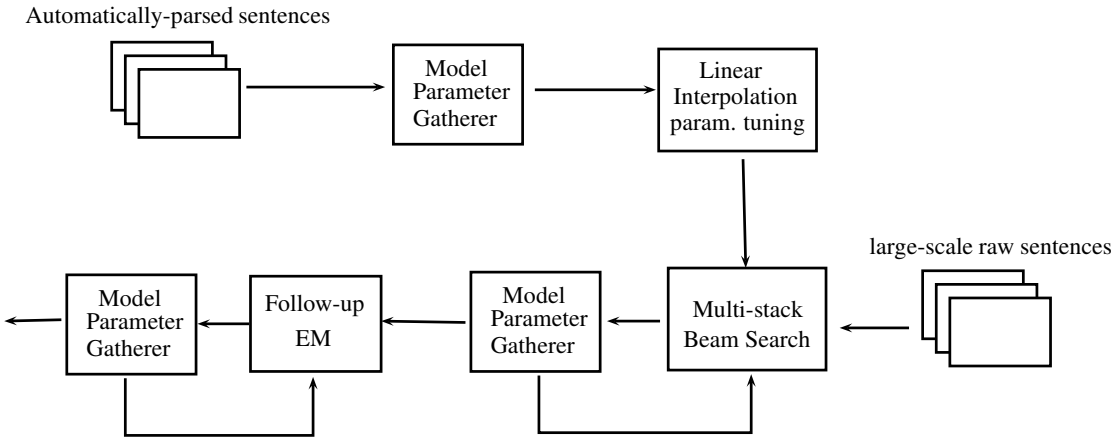

Figure 5.3: The work flow of the training process of the composite language model.

size N_l of the parsed trees generated by N-best list approximate EM, as well as the vocabularies of words, POS tags, non-terminal tags and CONSTRUCTOR options.

2. The initial parameters are initialized by a set of manually parsed sentences, eg. the 40k manually-parsed sentences of Treebank. One can also use a large amount of automatically-parsed sentences. This step is processed in parallel by the clients.

3. As discussed in Section 4.1.2, we perform EM algorithm for the standard PLSA model on training corpus $\mathcal{D}$, and then keep the τ most likely topics according to the values of $p(g|d)$, $\tau = 2$ or 5, the rest topics are purged.

4. The initial counts in the maximum order of each parameters, ie. $C(w, w_{-n+1}^{-1} h_{-m}^{-1} g)$, $C(t, wh_{-m}^{-1}.\text{tag})$ and $C(a, h_{-m}^{-1})$ are gathered by "Model parameter gatherer", from the clients to the servers. One may notice that "Model parameter gatherer" is actually a standard MapReduce procedure.

5. The EM algorithm is performed to estimate the λs of the generalized recursive linear interpolation scheme. Once the linear interpolation parameters are estimated, they will not be modified during the N-best list approximate EM and the follow-up EM.

6. The N-best list approximate EM is performed. For each sentence, there are N_l weighted parse trees generated. And the summation of the weights equals to one. Note that in this step, a large-scale set of raw English sentences can be involved. The counts of the new iteration for each component are estimated by the clients, while the parameters of the old iteration are stored in servers.

7. *Model parameter gatherer* gather the parameters for the new iteration from the clients to the servers. Repeat step 6 and 7 until convergence.

8. Fix the model parameters of CONSTRUCTOR, TAGGER and SEMANTIZER and do the follow-up EM algorithm, in order to re-tune the WORD-PREDICTOR.

9. *Model parameter gatherer* gathers the parameters for the new iteration from the clients to the servers. Repeat step 8 and 9 until convergence.

6

Rescoring Lattices from a Phrase based MT Decoder

In the previous chapters, we have already introduced a model structure and the training procedure of the proposed large scale distributed composite language models. In this chapter, we present a two-pass decoding strategy in order to integrate this model into a phrase-based machine translation system. Instead of the N-best list rescoring, one traditional way to involve a more sophisticated LM, we resort to an efficient A^* search procedure, which rescores the lattices generated by the first-pass decoder with a standard n-gram LM. Furthermore, since the composite LM is included as an additional feature, we use a simple but efficient tuning strategy for optimizing the feature weight.

6.1 Moses: A phrase-based translation system

In this section, we first briefly introduce Moses, a state-of-the-art phrase-based translation system. Moses implements the phrase-based decoder proposed by Koehn [Koehn et al., 2007] is a beam search multi-stack algorithm, similar to the stack decoder used in speech recognition. The English output sentence is generated from left to right in the form of partial translations. It starts with an initial empty hypothesis. A new hypothesis is expanded from an existing hypothesis by the translation of

a phrase as follows: A sequence of untranslated source words and a possible English phrase translation is selected. The English phrase is attached to the existing English output sequence. The foreign words are marked as translated and the probability cost (negative log-likelihood) of the hypothesis is updated. The cheapest final hypothesis with no untranslated source words is the output of the search.

The hypotheses are stored in multi-stacks where each stack contains all hypotheses in which the same number of source words have been translated. To reduce the number of hypotheses stored in each stack, Moses prunes out weak hypotheses based on the cost that combines the current cost and a future cost estimate, and keep only a beam of most promising hypotheses. The current cost is the total probability cost of the phrases that have been translated so far in the hypothesis which consists of the translation, distortion, and the language model probabilities. The future cost estimates for any sequence of consecutive foreign words can be computed through dynamic programming by ignoring the distortion cost and simply consider phrase translation and language model costs.

6.2 Two-pass decoding strategy

The large size of the search space in machine translation decoding sometimes makes it impossible to apply the most sophisticated models to the full space of translation hypotheses. For example, in the work of [Schwartz et al., 2011; Schwartz et al., 2011b], an incremental syntactic language model was directly integrated into the phrase-based decoder in Moses, and made a BLEU score reduction by 0.2∼0.3. However, as acknowledged in their paper, such integration drastically slowed down the decoding by about 2000 times, therefore only affordable to translate the sentences with less than 20 words. Therefore, there might be too much time-consumption if we directly integrate our composite LM into Moses in the first-pass decoder.

For this reason, it is more practical that we first generate a large subset of the most likely translations according to the first-pass decoder, and then rerank the hypotheses

with more sophisticated models. We refer this idea as "two-pass" decoding in machine translation. The search space in the second pass is much more restricted compared to the first pass so we can afford using better usually also computationally more intensive language models.

There are two most popular two-pass strategies: *N-best list rescoring* and *lattice rescoring*. They are different with respect to the number of intermediate hypotheses saved after the first pass and the form in which they are stored.

1. *N-best list rescoring*. A list of complete hypotheses along with simpler language model scores are retained and then rescored using more complex language models. One reason which makes the N-best rescoring framework attractive is the possibility to use "whole sentence" language models: models that are able to assign a score only to complete sentences due to the fact that they do not operate in a left-to-right fashion. However, due to the limited number of hypotheses in the N-best list, the second pass recognizer might be too constrained by the first pass so a more comprehensive list of hypotheses is often needed. Rescoring on N-best lists was significantly influenced by the language model used in the first-pass decoder and the size of the N-best list. The smaller list reduced the chance that a better hypothesis can survive the first-pass decoding, while the large N-best size might slow down the first-pass decoding.

2. *lattice rescoring*: In comparison to N-best list rescoring, rescoring on the lattice [Ueffing et al., 2002] serves as an efficient compromise between the simple N-best list rescoring and the direct integration in decoder. It is desirable for a lattice to have a set of the hypotheses, which is as complete as possible and not biased towards the model in the first-pass decoding and at the same time in a manageable size.

The intermediate format in which the hypotheses are stored in the lattice is a directed acyclic graph in which the nodes are a subset of the language model states in the

composite hidden Markov model and the arcs are labeled with words. The lattice of a simple sentence is illustrated in Figure 6.1.

We propose an effective A^* search lattice rescoring algorithm, which more effectively integrates their composite language model into a phrase-based decoder of Moses. A^* is appealing for our problem due to following two reasons:

1. The lattice can be conceptually structured as a prefix tree of hypotheses, which is consistent with the left-to-right generative pattern of the composite language model.

2. The algorithm operates with the whole prefix, making it ideal for incorporating language models whose memory is the entire prefix.

The lattices we work with are generated by the first-pass decoder using an n-gram language model. The proposed A^* search procedure is conducted, while simultaneously performing the multi-stack search for SLM model and the topic distribution update for PLSA model. During the A^* search, the current cost of a partial hypothesis is the weighted linear combination of features, while the future cost is heuristically estimated from the current ending node of the partial hypothesis to the ending nodes of the lattice. Moreover, since we introduce an additional language model in the system, we propose a lattice-based tuning strategy to re-optimize the model parameters.

6.3 A^* Search for lattice rescoring

Figure 6.1 gives a simple lattice for a sentence. [1] A lattice can be formulated as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. $\mathcal{N}$ is a set of nodes. Every node n_i has a number s_i, which corresponds to the stack where this node is created in the first-pass decoding [Koehn et al., 2003]. It actually equals to how many source words have been translated by the hypothesis on that node. $\mathcal{E} = \{l_1, l_2 \cdots\}$ is a set of directed links. Each of them

[1] The lattice denoted in our paper is referred as a "search graph" in Moses.

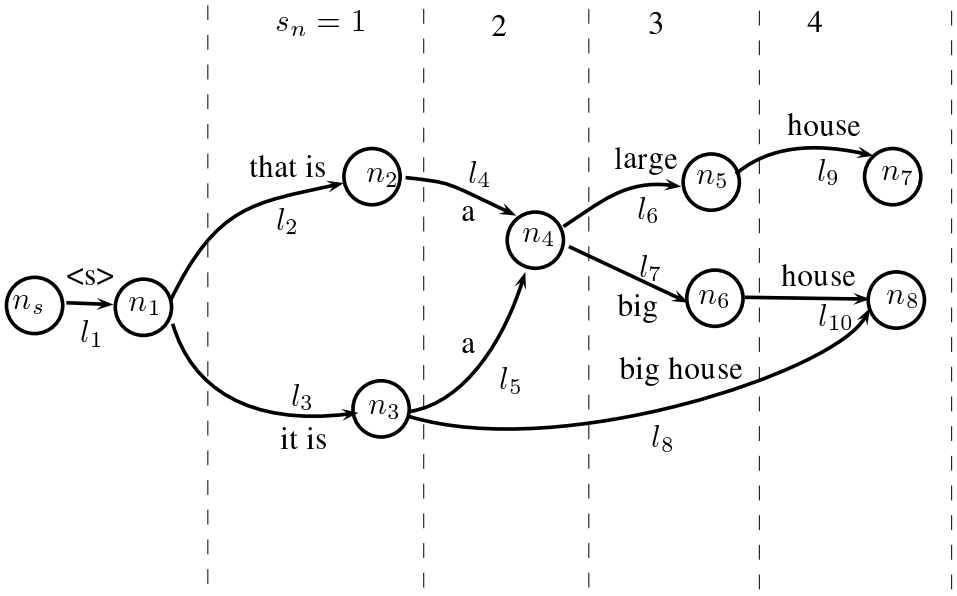

Figure 6.1: A lattice generated from Moses.

connects two nodes in $\mathcal{N}$ and represents a phrase in the target language. $\mathcal{G}$ is a directed acyclic graph (DAG), which means, for the starting node n_1 and the ending node n_2 connected by any directed link l_i, we have $s_1 < s_2$.

The A^* algorithm [Nilsson, 1971] is basically a tree search strategy that could be compared to depth-first traversal: pursue the most promising path as deeply as possible. All the hypotheses are organized as a prefix tree. We wish to obtain the maximum scoring hypothesis under the scoring function f.

$$h^* = \arg\max_{\bar{h}} f(\bar{h})$$

where

$$f(\bar{h}) = \sum_{l \in \bar{h}} \sum_{\alpha_k \in \underline{\alpha}} \alpha_k f_k(l)$$

$\underline{\alpha}$ is the feature weights, corresponding to one feature $f_k(l)$. Here, $\bar{h}$ is a complete hypothesis (there is no outgoing link at the ending node of $\bar{h}$). The A^* search tends to search the maximal solution without scoring all the possible hypotheses, if possible with a minimal computational effort.

To accomplish this, the A^* search defines a ranking score for the *partial hypothesis* h as $g(h) = f(h) + e(h)$, where $f(h)$ is the current feature score of the partial hypothesis, and the future cost estimation $e(h)$, is pre-computed before the lattice rescoring. We introduce the future cost in the next section.

Algorithm 1 A^* search for lattice rescoring

Require: A lattice $\mathcal{G}$, A^* stack size τ and the score threshold T

1: h_{top}, $h \leftarrow \{\}$
2: Stack $S \leftarrow \{\}$
3: Insert(h,S)
4: **repeat**
5: $h_{top} \leftarrow$ Pop(S)
6: **for** $l \in$ OutgoingLink of h_{top} **do**
7: $h \leftarrow$ Expand(h_{top},l) ▷ extend h along l
8: Insert(h,S) ▷ S ranks the elements by $g(h)$
9: Prune S according to $g(h)$ with τ and T
10: **end for**
11: **until** h_{top} is complete ▷ h_{top} has no outgoing links.
12: **return** h_{top}

The A^* search algorithm for lattice rescoring is described in Algorithm 1. The algorithm maintains a stack S, whose elements are the partial paths ranked by $g(h)$. The stack S has two variables: (i) A limit stack size τ. Any partial paths ranked outside of the size will be pruned. (ii) A threshold T. Any h, whose ranking score is lower than the top-1 element in S by more than T, is pruned. The algorithm repeatedly pops the top-1 hypothesis and expand it along every outgoing link l. A newly generated hypothesis h' obtain $f(h')$ by adding $f(h)$ with the weighted feature

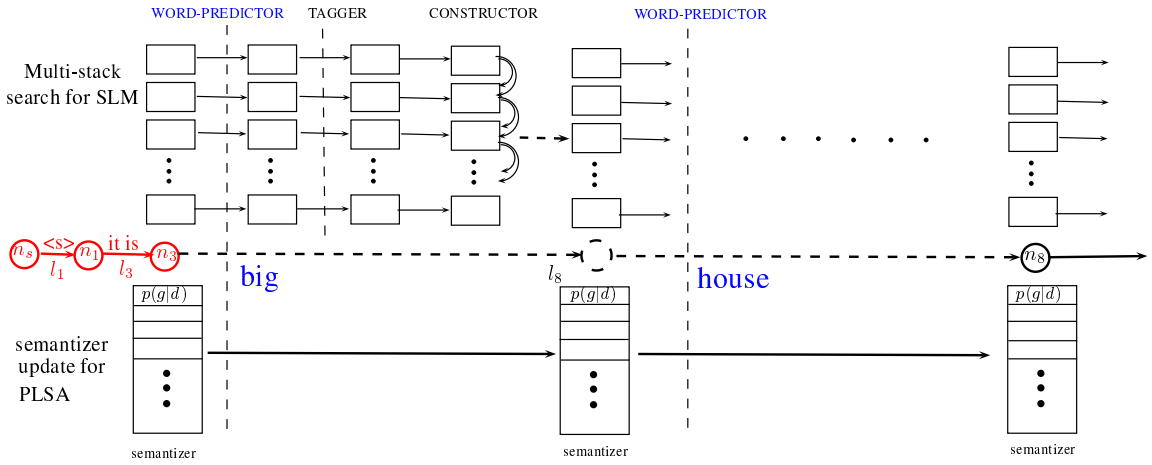

Figure 6.2: The partial hypothesis contains l_1 and l_8. As new words are predicted, the topic distribution for PLSA SEMANTIZER and the stacks for the multi-stack search for SLM is simultaneously updated.

score of l.

$$f(h') = f(h) + \sum_{\alpha_k \neq \alpha_{clm}} \alpha_k f_k(l) + \alpha_{clm} \cdot score_{clm}(l|h) \quad (6.1)$$

where the composite LM score $score_{clm}(l_i|h) = \sum_{w \in l} \log P(w|W_{(h,w,l)}, d)$. We calculate the log probability (Eq. 4.23) of every w in l sequentially, based on its history $W(h, w, l)$, which includes not only the words in h, but also the previous words in l. Other features can be regarded as constants. Each generated hypothesis h' is inserted back into S, while S prunes them with τ and T, until there is no outgoing links for the top-1 hypothesis. Finally, we finish the second-pass decoding by returning the top-1 hypothesis.

To compute $score_{clm}(l|h)$, it is necessary to simultaneously update the multiple stacks for multi-stack search of SLM model and the SEMANTIZER of PLSA model. We present the expansion of an partial hypothesis along l (Line 7 in Algorithm 1) by a simple example shown Figure 6.2. The part in red is a hypothesis $h = \{l_1, l_3\}$. This is a zoom-in of Figure 6.1. The hypothesis h has generated the words "<s> it is", and

l_8 contains the phrase "big house". The expanded partial hypothesis $h' = \{l_1, l_3, l_8\}$. Different from [Chelba and Jelinek, 2000] for speech recognition, a link of the lattice in this paper represents a phrase, instead of a single word. Therefore, as shown in Figure 6.2, $score_{clm}(l_8|h)$ is computed by two steps, for "big" and "house" respectively. For each word, following steps are taken,

- $P(w|W_{(h,w,l_8)}, d)$ is firstly calculated by WORD-PREDICTOR, blue in Figure 6.2.
- TAGGER assigns w_k a POS tag. Then, the partial parse is repeatedly built up by CONSTRUCTOR. These procedures are similar to the standard SLM model [Chelba and Jelinek, 2000].
- SEMANTIZER updates each $p(g|d)$ by Eq. 4.25.

After computing $score_{clm}(l_8|h)$, the current cost $f(h')$ is updated by Eq. 6.1.

6.3.1 Future cost estimation

The future cost of $e(h)$ is an estimation from the current ending node of the partial hypothesis h, denoted as n_{ending}, to the ending nodes of the lattice. For speech recognition, [Chelba and Jelinek, 2000] assumed that the future cost consists of the acoustic model along this path and the composite language model probability of the words along this path which is approximated by n-gram plus a compensation term $logP_{comp}$. The expected value of this compensation term can be heuristically estimated from the difference in perplexity between n-gram and the advanced LM.

In this paper, we propose a heuristic-driven alternative for the future cost estimation. Although the composite language model score for a link l, given the previous partial hypothesis h, replies on types of global information, such as the partial parse T_k and the topic distribution $p(g|d)$, a rough estimation of the future cost can treat the composite LM as a local feature.

A Viterbi [Viterbi, 1967] forward pass is performed from the nodes labeled by s_1 to the nodes with maximal s. For each link, we calculate the estimated composite LM

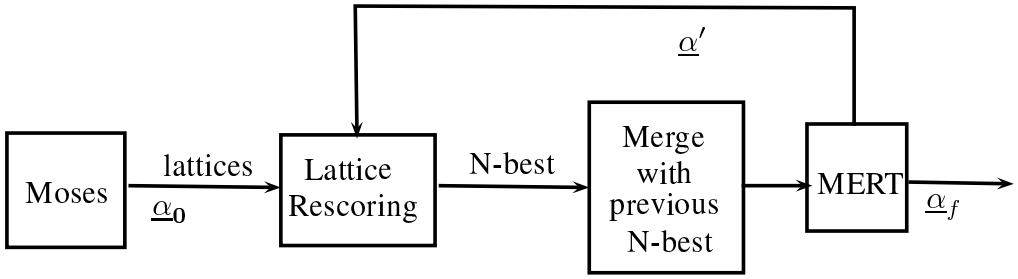

Figure 6.3: The tuning strategy

score by the history of the top-1 path of its incoming node by dynamic programming. In this case, given a lattice $\mathcal{G}$, the future cost is completely determined by the identity of each node; a Viterbi backward pass over the lattice can store at each node the corresponding value of $e(h) = e(ending\_node(h))$, for any h ends at this node, such that it is readily available before the A^* search. Due to the strong pruning of the multi-stack search and the limited beam size of the A^* search, the proposed A^* search procedure is not *admissible*, and thus cannot guarantee to find the optimal solution.

6.4 The tuning strategy

We apply the composite language model as an additional feature into the lattice rescoring procedure. Thus, we also exploit a lattice-based tuning strategy in order to re-estimate the weight of the features. In [Macherey et al., 2003], MERT [Och, 2003] algorithm has been extended directly to the lattices. However, if the large scale distributed composite language model is provoked in this lattice-based MERT, the time consumption is expected to be huge. Therefore, we provide a simple but efficient framework for optimizing the parameters:

1. The tuning set is iteratively decoded by Moses phrase-based decoder with a standard n-gram model and the model parameters are repeatedly tuned by MERT until convergence. Then we can obtain the initial parameters $\underline{\alpha}_0$ for our work. We obtain the lattices for each sentence and also append one dimension to $\underline{\alpha}_0$ for

composite LM, and set up its value as zero.

2. With the current model parameters $\underline{\alpha}$, we iteratively perform the lattice rescoring procedure, described in Section 6.3 and generate the N-best list for each sentence. We discuss the generation of N-best list in section 6.5. We also merge the N-best lists with the one of previous iteration to increase the searching space.

3. Tune the model parameters $\underline{\alpha}$ by MERT based on the merged N-best list.

4. Repeat 2 and 3 until convergence.

5. Then use the final parameters $\underline{\alpha}_f$ for testing.

The experiments show that the tuning strategy cannot be simply substituted by replacing the n-gram model in the first-pass decoder with the composite LM, meanwhile keeping the LM weight unchanged. Typically, we set up the N-best list sizes as 500 for the MERT both in the first-pass and second-pass decoding. In testing, we use $\underline{\alpha}_0$ for the first-pass decoding and lattice generation, and use $\underline{\alpha}_f$ for lattice rescoring.

6.5 Implementation Issues

As shown in section 6.3, an N-best list per sentence need to be generated for the tuning strategy. To do this, we slightly modify Algorithm 1 at Line 11: the A^* search will not halt until we find h_{top}, which is complete, by N times.

In the previous sections, we in default perform the lattice rescoring algorithm sentence by sentence. When we generate the N-best list, for the first source sentence in a document, prior to the rescoring procedure, we assign the initial topic distribution $p(g|d)$ as the one of the training corpus, which is a byproduct of training the composite LM. For every rest sentence of that document, we use the final topic distribution from the translation result of the last-translated sentence. However, we use another mechanism to generate $p(g|d)$ during the A^*search for a testing sentence: we connect

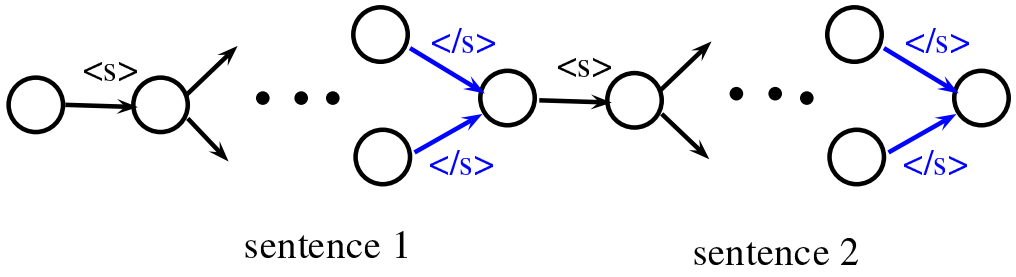

Figure 6.4: Multiple lattices are combined for testing.

the lattices from the first sentence to the last sentence in order to form one single large lattice, shown in Figure 6.4, such that the SEMANTIZER $p(g|d)$ is estimated without empirical selection of the previous sentence's translation.

Similar to Section 5.2, we use a client-server mechanism with message passing interface (MPI) to exploit the distributed composite language model. The language model is hashed into a number of servers according to two elements in history, the last n-gram word and the topic g, since in this way the language model can be evenly loaded in the servers. On the client side, we split the test set by documents, because it is necessary to keep the semantic content on the documental level.

7

Perplexity Experiments

In this section, we first explain the experimental set-up of language models for our experiments, and then show comprehensive perplexity results in various situations.

7.1 Experimental set-up

In previous works [Bellegarda, 2000; Charniak, 2001; Chelba and Jelinek, 2000; Chelba, 2000; Gildea and Hofmann, 1999; Roark, 2001], all complex language models are trained in relatively small data sets. Therefore, there is an impression that complex language models only lead to better results than n-grams on small training corpus. For example, on page 482 in [Jurafsky and Martin, 2008], the authors 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."

7.1.1 Corpora selection

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 other 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 non-verbalized punctuation and we remove all punctuation for the perplexity experiments. Table 7.1 gives the detailed information on how these data sets are chosen from the LDC English Gigaword corpus (LDC2009T13).

7.1.2 Vocabularies and n-gram statistics

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 occurred words in 44 millions tokens corpus.
- POS tag (also TAGGER operation) vocabulary: 69, closed - no TAGGER operation outside of this vocabulary is allowed.
- Non-terminal tag vocabulary: 54, closed.
- CONSTRUCTOR operation vocabulary: 157, closed.

The out-of-vocabulary (OOV) on 44 million, 230 million, 1.3 billion tokens training corpus is 0.6%, 0.9% and 1.2% respectively, the OOV on 1.7 million and 13.7 million tokens check corpus is 0.6% and 1.3% respectively, finally the OOV rate on 354K tokens test corpus is 2.0%.

Table 7.2 gives the statistics about the number of types of n-grams on these three corpora.

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
afw	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
354k tokens test corpus	
cna	20041101.0006 ∼ 20041217.0009

Table 7.1: 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.

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

Table 7.2: 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.

7.1.3 Model initialization

SLM initializes its model parameters by human-parsed sentences in Penn Treebank corpora. Similar to SLM [Chelba and Jelinek, 2000; Chelba, 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 [1] 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; 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.

7.2 Perplexity Results

7.2.1 Baseline results

Table 7.3 gives the perplexity results [Jelinek et al., 1977] of n-grams (n=3, 4 and 5) using linear interpolation and Kneser-Ney smoothing when training corpus has 44 million, 230 million, and 1.3 billion tokens respectively.

We have implemented distributed n-gram with linear interpolation smoothing, but we don't have distributed n-grams with Kneser-Ney smoothing implemented by our own. Instead, we use SRI Language Modeling Toolkit (SRILM) to obtain perplexity

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

44M	LINEAR	KNESER-NEY (PPL reduction)
n=3	262	244 (6.8%)
n=4	258	235 (8.9%)
n=5	260	235 (9.6%)
230M	LINEAR	KNESER-NEY
n=3	217	195 (10.1%)
n=4	200	183(8.5%)
n=5	201	183(8.9%)
1.3B	LINEAR	KNESER-NEY
n=3	161	—
n=4	141	—
n=5	138	—

Table 7.3: Perplexity results of n-grams (n=3, 4 and 5) using linear interpolation and Kneser-Ney smoothing when training set is 44 million, 230 million or 1.3 billion tokens corpus respectively.

results of n-grams with Kneser-Ney smoothing for 44 million and 230 million tokens corpora using a single machine at Ohio Supercomputer Center that has 20G memory. We are not able to have perplexity results of n-grams with Kneser-Ney smoothing on 1.3 billion tokens corpus, thus we leave these results blank in Table 7.3.

From the results in Table 7.3, 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. We can observe that Kneser-Ney smoothing obtains the reduction of Perplexity as less than 10%, in comparison to the linear interpolation smoothing.

Corpus	n	# of TOPICS	PPL	TIMES (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	280	100	2.280B
230M	4	5	146	25.6	400	100	0.681B
1.3B	5	2	111	26.5	400	100	1.790B
	5	5	102	75.0	400	100	4.391B

Table 7.4: 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.

7.2.2 Experimental results of n-gram/PLSA

We have to keep only a small set of topics due to the consideration in both computational time and resource demand. Table 7.4 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.

7.2.3 Experimental results of n-gram/PLSA/SLM

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 7.5 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. Table 7.6 gives the statistics about the number of types in the predictor of the m-SLMs on these three corpora, where for 230 million tokens and 1.3 billion tokens corpora, we cut off its fractional expected counts that are less than a threshold 0.005, to significantly reduce the number of predictor's types by 70%. Compared to the n-gram token sizes presented in Table 7.2, SLM become much larger even after a heavy pruning.

In Table 7.5, 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

LANGUAGE MODEL	44M n=3,m=2 (PPL reduction)	44M n=3,m=2 (PPL reduction)	44M n=3,m=2 (PPL reduction)
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/SLM+ m-SLM/PLSA	183 (30.2%)	139 (30.5%)	94 (31.9%)
n-GRAM/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 7.5: 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. () is the PPL reduction compared to the baseline n-gram in linear interpolation smoothing.

	$m = 2$	$m = 3$	$m = 4$
44M	189,002,525	269,685,833	318,174,025
23M	267,507,672	1,154,020,346	1,417,977,184
1.3B	946,683,807	1,342,323,444	1,849,882,215

Table 7.6: Statistics about the number of types in the predictor of the m-SLMs ($m = 2, 3, 4$) on 44 million, 230 million and 1.3 billion tokens corpora. For 230 million and 1.3 billion tokens 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%.

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. We have checked that the conditional probability of every composite language model sums to 1 for a 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 PLSA component, but when adding SLM component into n-gram/PLSA, there is a further 10% relative perplexity reduction.

Table 7.7 shows how big the composite 5-gram/PLSA, 5-gram/2-SLM (or 2-gram/4-SLM), and 4-SLM/PLSA are when trained by 1.3 billion tokens 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 is respectively 400, 240 and 400, and the number of clients to store partitioned data of 1.3 billion tokens corpus is 100 for these three composite language models. There is no way to store the parameters of the linear combination of

COMPOSITE MODELS	# OF TYPES	# OF SERVERS	# OF CLIENTS
5-gram/PLSA	4.39B	400	100
5- GRAM /2-SLM 2 GRAM /4-SLM	2.01B	240	100
4-SLM/PLSA	4.88B	400	100

Table 7.7: Counts of the types in predictor of 5-gram/PLSA, 5-gram/2-SLM (or 2-gram/4-SLM), and 4-SLM/PLSA when trained on 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%.

the composite 5-gram/PLSA, 5-gram/2-SLM (or 2-gram/4-SLM), and 4-SLM/PLSA in our current available supercomputer resources.

Table 7.8 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 use *one step online EM*, *online EM with fixed learning rate* and *batch EM* respectively. The online 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 1.3 billion tokens corpus, we have to shrink the order of n-gram and m-SLM in order to store them in a supercomputer having 1000 cores. 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. For composite 5-gram/4-SLM model or its linear combination with models, we in fact use its approximation, a linear combination of 5-gram/2-SLM and 2-gram/4-SLM.

Next, we conduct experiments where we fix the size of training data and increase the complexity of our language models. Since available resources are limited to prevent us to consider complex language models that are trained on 1.3 billion tokens

Language model	44M $n = 3, m = 2$ (Reduction)	230M $n = 4, m = 3$ (Reduction)	44M $n = 5, m = 4$ (Reduction)
n-GRAM (LINEAR)	262	200	138
n-GRAM /PLSA	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	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	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%)

Table 7.8: Perplexity results for composite n-gram/PLSA and n-gram/m-SLM/PLSA language models on test corpus, where + denotes linear combination, / denotes composite model; n is the order of n-gram and m is the order of SLM, and superscripts 1, 2, 3 denote using one step online EM, online EM with fixed learning rate and batch EM during testing respectively.

LANGUAGE MODEL	44M n=3,m=2 (PPL reduction)	44M n=4,m=3 (PPL reduction)	44M n=5,m=4 (PPL reduction)
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/SLM+ m-SLM/PLSA	183 (30.2%)	178 (31.0%)	180 (30.8%)
n-GRAM/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%)

Table 7.9: 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. () is the PPL reduction compared to the baseline n-gram in linear interpolation smoothing.

corpus, we consider complex language models trained on 44 million tokens corpus instead. Table 7.9 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, however, when we increase the order for n-gram and m-SLM to $n = 5$ and $m = 4$, 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.

Finally, we compare our composite language model with Recurrent neural network based language model (RNNLM), proposed in [Tomas et al., 2010; Tomas, 2012]. Instead of combining the language models which describe lexical, syntactic and semantic language models, RNNLM used an architecture that is usually called a simple recurrent neural network to integrate the context. We do the comparison on the 44M level corpus. The RNNLM gives a perplexity of 248 , while it obtain a perplexity of 184 when combined with tri-gram with KN-smoothing. In Figure 7.9, the best composite language model give a perplexity of 176 under "n-GRAM/m-SLM/PLSA". So, our model is moderately better than RNNLM.

Moreover, the composite language model has advantages on two other aspects: (1) In RNNLM, the effect of the context is only reflected by a black box in NN, while the composite LM directly reflect the combination on the condition of the word-predictor. (2) No distributed training strategy has been proposed yet for a large scale training corpus.

7.2.4 A specific example

We show an example of sentence probability that is provided by 5-gram, 5-gram/PLSA, and 5-gram/4-SLM+5-gram/PLSA respectively, these language models are trained using 1.3 billion tokens corpus. We would like to demonstrate that our composite model is able to extract topic information and grammatical structure to improve word prediction for natural language.

We choose 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 below the word conditional probabilities given its document history for the 4th sentence. The first line is the 4th sentence, the second line marked by a denotes the natural log value of the conditional word probabilities given its document history computed by 5-gram, the third line marked by b denotes the natural log value of the conditional word probabilities given its document history computed by 5-gram+PLSA, the forth line marked by 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 marked by d denotes the natural log value of the conditional word probabilities given its document history computed by 5-gram/PLSA, and the sixth line marked by 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 -7.6000 -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

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

Table 7.10: Statistics when n-grams are the same as SLM's WORD-PREDICTOR in the most likely parse structure of each sentence in training corpora.

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*, *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 topic to influence word prediction. In contrast, there is no affect 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 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, n-gram has to increase its order to 11 or 8. Linear combination model 5-gram+4-SLM+PLSA is quite effective, however, it has negative impact for the prediction of function words such as *of the* after the word(s) *natural* or *political bureau*.

Table 7.10 shows the statistics when n-grams are the same as 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, SLM component will be effective to furnish sentence level long range grammatical information.

This example and Table 7.10 clearly demonstrate that n-gram alone is not viable to achieve similar affect that SLM and PLSA can supply 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.

7.3 Discussion

7.3.1 Why are composite LMs better than linear interpolation?

To better explain and analyze our model, we mark the perplexity results for 40 million tokens corpus in Table 7.5 on the vertices in Figure 3.3 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 $p(w|g)$, 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:

- valuable relative frequency estimates such as $f(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$, $f(w|w_{-2}w_{-1}h_{-2}h_{-1})$ etc. are encoded into the composite language model as seen from Figure 3.3. As long as there are events such as $w_{-2}w_{-1}wh_{-2}h_{-1}g$ etc. occur explicitly or im-

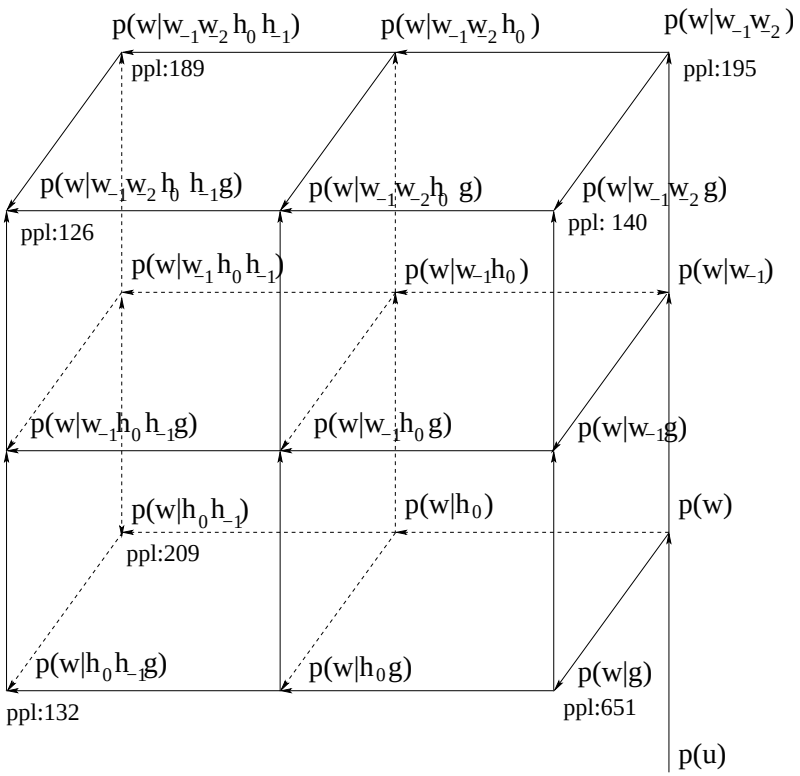

Figure 7.1: Recursive linear interpolation lattice to estimate WORD-PREDICTOR $p(w|w_{-2}w_{-1}h_{-1}h_0g)$ of the composite trigram/2-SLM/PLSA language model, where $\mathcal{U}$ is the vocabulary in which the predicted random variable w takes values and $p(\mathcal{U})$ denotes uniform distribution of $\mathcal{U}$. The lattice is formed by three linear Markov chains, $w_{-2}w_{-1}$, $h_{-1}h_0$ and g. Starting from $p(\mathcal{U})$, each vertex is visited in a bottom up, back to front, and right to left order.

plicitly in training corpus, the composite trigram/2-SLM/PLSA will take them into account to improve the prediction power for test data, while linear combination of trigram, 2-SLM and PLSA just neglects a large amount of these valuable information.

- the weights used in simple linear combination are context independent, thus more restricted. Similarly, the composite trigram/2-SLM/PLSA language model is more powerful and expressive than linear combination of pairwise composite language models, e.g., trigram/2-SLM, trigram/PLSA, and 2-SLM/PLSA, since composite trigram/2-SLM/PLSA can take advantage of relative frequency estimate $f(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$, $f(w|w_{-2}w_{-1}h_{-1}g)$, and $f(w|w_{-1}h_{-2}h_{-1}g)$. However, the improvement in this case shrinks since pairwise composite language models use some valuable lower order relative frequency estimates such as $f(w|w_{-2}w_{-1}g)$ etc. Put it in another way, each vertex of the lattice in Figure 3.3 is an expert of WORD-PREDICTOR who is proficient in making prediction based on the context represented at the vertex, it predicts word 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, the WORD-PREDICTOR of the composite trigram/2-SLM/PLSA, i.e., $p(w|w_{-2}w_{-1}h_{-2}h_{-1}g)$, oversees all available information to make the most powerful prediction.

7.3.2 Does "more data" play more important role?

To summarize, as a sub-problem for machine translation 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, there is always a trade-off between approximation error and estimation error [Barron and Sheu, 1991], reminiscent of "bias-variance" trade-off for a regression problem with a quadratic cost function [Hastie et al., 2009]. Figure 7.2 explains

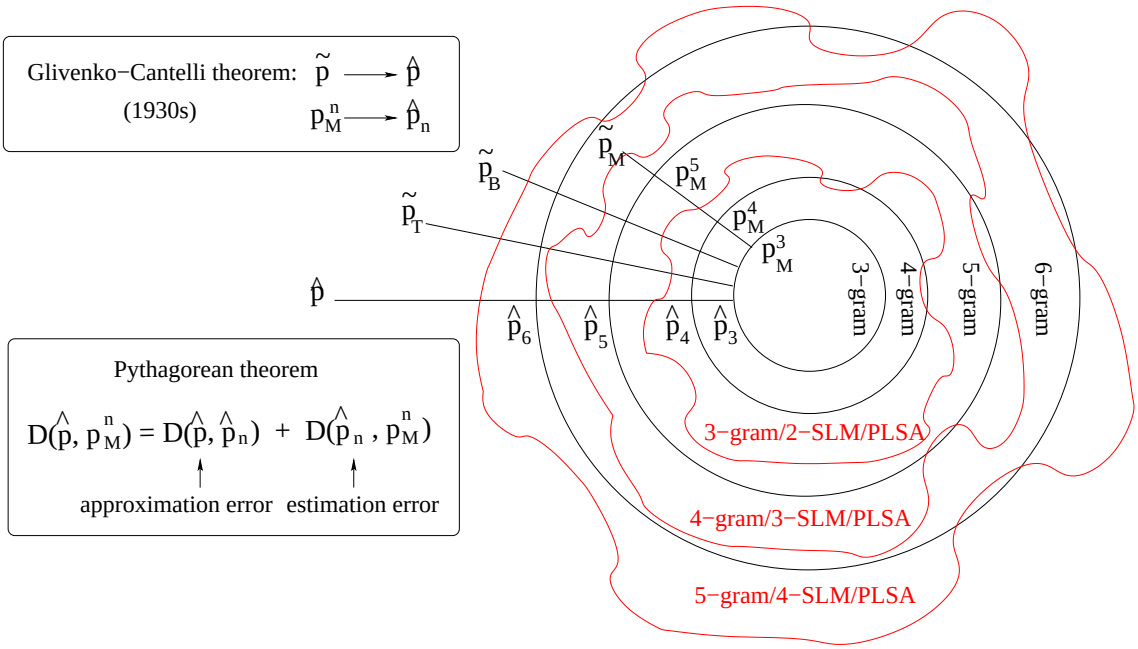

Figure 7.2: Language modeling is a data rich and feature rich density estimation problem, the information projection from true distribution and empirical distribution to n-grams is unique, while the information projection from true distribution and empirical distribution to composite language models might be local optimal. There is a trade-off between approximation error and estimation error for composite language models.

the perplexity results in Tables 7.3 and 7.5 from 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 et al., 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 million tokens corpus, $\tilde{p}_B$ denotes the empirical distribution for billion tokens corpus, and $\tilde{p}_T$ denotes the empirical distribution for trillion tokens corpus. The information projection of $\tilde{p}_M$ to trigram is p_M^3, to 4-gram is p_M^4, and to 5-gram is p_M^5. 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 available. The distance between $\tilde{p}_M^n$ and $\hat{p}_n, n = 3, 4, 5$, $D(\tilde{p}_M^n, \hat{p}_n)$, is the estimation error when only million tokens 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 [Amari and Nagaoka, 2000; Barron and Sheu, 1991; Wang et al., 2009]. In language modeling research, since $\hat{p}$ is unknown, the distance between $\hat{p}$ and $p_M^n, n = 3, 4$ is approximately computed by 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 project of empirical distribution on n-gram converges to the information project on n-gram of true distribution, i.e., the estimation error shrinks to 0. Similarly, we can define the information projection of $\hat{p}$ or $\tilde{p}$ to the composite language models, and corresponding approximate error and estimation error etc. In this case, Pythagorean theorem breaks down due to the non-convexity of the set of composite language models. 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. Also, see [Chelba et al., 2010]. This, as well, is true for our composite language models as justified from the results in Tables 7.5 and 7.9: the composite n-gram/m-SLM/PLSA language model has rich features thus has smaller approximation error than n-gram, m-SLM, PLSA, or any composite model of two, or their linear combinations. Table 7.5 shows that the information project of the empirical distribution for million and billion tokens corpus, $\tilde{p}_M$ 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.

8

Machine Translation Experiments

In this chapter, we report the results when we apply the composite language model to the task of reranking the N-best list generated from a state-of-the-art parsing-based machine translation system for a Chinese-to-English translation. Next, experiments on the French-to-English task using WMT2008 and WMT2010 datasets prove that the A^*-based lattice rescoring is more effective to show a significant predominance of the established composite language model over the n-gram model than the traditional N-best list rescoring.

8.1 N-best reranking experiments

8.1.1 Results on BLEU score

We have applied our composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA[1] language model that is trained by 1.3 billion word corpus for the task of reranking the N-best list in statistical machine translation. We used the same two 1000-best lists that were used by Zhang et al. [Zhang et al., 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 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 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, 9 pieces are used as training data to optimize the BLEU score by MERT, a remaining single piece is used to rerank 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. The mean and variance of the BLEU score are calculated with each different LMs. We assume that the score follows Student's t-distribution and we compute the confidence interval (CI) with confidence of 95% according to mean and variance. Table 8.1 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% BLEU score improvement over the baseline and 0.79% BLEU score improvement over the 5-gram. We are not able to further improve BLEU score when we use either 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 1000-best list, and essentially only $20 \sim 30$ distinct sentences are there in the 1000-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 rerank the 1000-best list and obtain the BLEU score. To obtain the confidence interval of the BLEU score, we resort to bootstrap resampling described by Koehn [Koehn, 2004]. We randomly select 10 reranked documents from the 20 reranked 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 1000 times. When we compute

SYSTEM MODEL	MEAN (%)	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

Table 8.1: 10-fold cross-validation BLEU score results for the task of reranking the 1000-best list generated on 919 sentences of 100 documents from the MT03 Chinese-English evaluation set.

SYSTEM MODEL	MEAN (%)	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

Table 8.2: BLEU score results for the task of reranking the 1000-best list generated on 191 sentences of 20 documents from the MT04 Chinese-English evaluation set.

the confidence interval with 95% confidence, we drop top 25 and bottom 25 BLEU scores, and only consider the range of 26th to 975th BLEU scores. Table 8.2 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% BLEU score improvement over the baseline. The composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA[1] language model gives 1.19% BLEU score improvement over the baseline and 0.68% BLEU score improvement over the 5-gram.

Chiang [Chiang, 2007] studied the performance of machine translation on Hiero, the BLEU score is 33.31% when n-gram is used to rerank the N-best list, however, the BLEU score becomes significantly higher 37.09% when the n-gram is embedded

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

Figure 8.1: Results of "readability" evaluation on 919 translated sentences of 100 documents, P: perfect, S: only semantically correct, G: only grammatically correct, W: wrong.

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 should result in much improved BLEU scores.

8.1.2 Experiments on readability

Besides reporting the BLEU scores, we look at the "readability" of translations similar to the study conducted by Charniak et al. [Charniak et al., 2003]. The translations are sorted into four groups: good/bad syntax crossed with good/bad meaning by human judges, see Table 8.1. 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. [Charniak et al., 2003] integrated Charniak's language model with the syntax-based translation model Yamada and Knight proposed [Yamada and Knight, 2001] to rescore a tree-to-string translation forest, whereas we use only our language model for N-best list reranking.

In Appendix A, we give examples of "perfect" sentences, or "only semantically correct" sentences, or "only grammatically correct" sentences.

Also, in the same study in [Charniak et al., 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.

8.2 Lattice rescoring experiments

In order to comprehensively evaluate the effectiveness of the proposed lattice rescoring mechanism as well as the tuning strategy, we conduct a series of experiments on a French-to-English translation task. In this section, our intention is to clarify following issues: (i) Our lattice rescoring approach does significantly assist the established composite language model to achieve much better performance than the standard n-gram model, which can not be acquired by a simple N-best list rescoring. (ii) The tuning strategy based on the lattice rescoring is equally important to guarantee the effectiveness of the composite LM.

8.2.1 Experimental set-up

We used the composite language model using two different data sets discussed in the previous chapter: one with 44 million tokens, and the other with about 230 million tokens. Both of them are chosen from the LDC English Gigaword corpus shown in Table **??**. We also set up an independent *dev* set, in order to tune the smoothing parameters used in the LMs.

Similar to the experiments in Chapter 7, we implement our lattice rescoring algorithm and the distributed language model in C++ with MPI, and conduct the

experiments in a supercomputing environment, in which we use at most 40 eights-core nodes for clients and servers.

Based on the two datasets, we construct four distributed composite language models, which vary from the types of involved basic LMs. Important statistics for these four language models are illustrated in Table 8.3, including the training token size, the vocabulary size, the n-gram order (consider $n-1$ words in history), the topic size and SLM headword order m (consider m headwords in history) and the processor number each LM uses. In lattice rescoring experiments, we expand the vocabularies from 60k in PPL experiments to $100k$, and we also extend the n-gram order of composite LM on the 230m dataset to 5-gram. CLM_{all}^{44}, which is trained on smaller corpora, is restricted to the combination of tri-gram, two headword SLM and 50-topic PLSA, while CLM_{tn}^{44} removes the SLM and only keeps the n-gram model and PLSA model. In contrast, we use the larger corpus to train CLM_{all}^{230} and CLM_{tn}^{230} and extend the order of the language model. Moreover, we increase the number of topics from 50 to 200 when the training token size is increased from 44 million to 230 million. We train CLM_{tn}^{44} and CLM_{tn}^{230}, which only include PLSA and n-gram, because we observe that the combination of PLSA and n-gram generally contributes the majority of the PPL reduction. Both of CLM_{all}^{44} and CLM_{all}^{230} are the composite models, which completely combined n-gram, SLM and PLSA under direct Markov random fields, without any linear interpolation with binary-combined language models.

For the baseline LMs, we construct two n-gram language models for the first-pass decoding in Moses, nLM^{44} and nLM^{230}, respectively for 44-million-tokens/3-gram and 230-million-tokens/5-gram. The baseline LMs are trained by SRILM [Stolcke, 2002], whose smoothing technique is modified Kneser-Ney smoothing [Chen and Goodman, 1996]. Although there is a difference between the smoothing method for the composite LMs and the baseline LMs, a preliminary experiment by lattice rescoring, shows that replacing the baseline n-gram LMs with their counterparts with linear interpolation leads to no significant change on the final results.

	CLM_{all}^{44}	CLM_{tn}^{44}	CLM_{all}^{230}	CLM_{tn}^{230}	nLM^{44}	nLM^{230}
token #	44 million	44 million	230 million	230 million	44 million	230 million
vocab. #	35.1k	35.1k	97.4k	97.4k	35.1k	97.4k
n-gram	3-gram	3-gram	5-gram	5-gram	3-gram	5-gram
topic #	50	50	200	200	–	–
SLM	2 headwords	–	2 headwords	–	–	–
server #	80	30	200	128	1	1

Table 8.3: Statistics for the language models we build. The first four are composite models, and the last two are baseline LMs . The vocabulary is open, which means any word out of the vocabulary is projected to <unk>.

The French-to-English translation task is conducted on the data sets of WMT2010 [1]. We obtained the phrase-based translation model from 1.8 million French-English sentence pairs provided by Europarl dataset. A problem one needs to be aware of is that the initial parameters of structured language model is trained from a set of parse trees, which are automatically parsed by OpenNLP. The paring models are trained from Treebank. Therefore, we modified the preprocessing scripts of Moses on the English side to make it consistent with the one of Treebank, while leaving the French-side preprocessing script unchanged.

We use news-test-2008 dataset (2051 sentences/90 documents) in WMT2010 as the tuning set and news-test-2010 dataset (2439 sentences/119 documents) as the test set. Following Moses' default setting on features, apart from the baseline language model, thirteen other features are included during the decoding in Moses, including seven reordering features, five translation model features and a word penalty. After the original fourteen parameters are tuned by MERT with the tuning set, we obtain the lattices of the tuning set and the test set by the tuned parameters. We observe that the average numbers of nodes and links in the lattice of each sentence is about

[1]The data sets are available at http://www.statmt.org/.

6.5k and 64.7k. It suggests a huge search space, sufficient enough for our algorithm to search for a better solution.

Moreover, there are a number of configurations involved in the lattice rescoring procedure and the tuning strategy. We set up the stack size τ of the A^* search as 500, and the threshold T is set to 3, which is the maximum difference of the ranking scores between the top-1 hypothesis and the bottom hypothesis in the stack. For each iteration of the tuning period, a 500-best list for each sentence is generated from the lattice rescoring. Finally, we set the tuning iterations with 30, when there is no significant improvement on BLEU. We use the parameters on the best iteration in tuning for testing.

8.2.2 Lattice rescoring results

We first evaluate the composite language model on the perplexity (PPL) reductions. We directly use the English reference of the tuning set and the test set to be the testing dataset for perplexity experiments. Table 8.4 shows that the composite language models can significant outperform the corresponding n-gram model with the same order. When only PLSA and n-gram are combined, such as CLM_{tn}^{44} and CLM_{tn}^{230}, the PPL reduction rate is 15$\sim$18%, and when structured language models are also involved, the reduction rate is increased to over 21$\sim$25%.

In Table 8.5, we report the results in two groups, one for LMs with 44 million training tokens, and the other one for LMs with 230 million tokens. For each group, we first list the results obtained from the Moses first-pass decoder. Then, we show the performance when different LMs are included as an additional features by N-best list rescoring. Again, we tune the parameters on the N-best lists of the tuning set before we perform testing. Next, we compare them with our lattice-rescoring algorithm. "$_{\text{-notune}}$" in Table 8.5 means we directly replace the language model score from the first-pass decoder, while keeping the feature weights unchanged, so we do not perform the tuning step in this case.

	newstest2008	newstest2010
token size	51.6k	64.6k
nLM^{44}	410	355
CLM_{tn}^{44}	335	291
CLM_{all}^{44}	**312**	**261**
nLM^{230}	208	178
CLM_{tn}^{230}	170	147
CLM_{all}^{230}	158	131

Table 8.4: Perplexity results for references.

Table 8.5 shows a significant improvement on the BLEU score caused by the composite LMs with the proposed lattice rescoring algorithm and the tuning strategy. From the table, we can have following observation. Firstly, when we include the composite language models, the traditional N-best reranking can only improve the BLEU by about 0.3% compared to the baseline n-gram models, while our lattice rescoring mechanism with the tuning strategy can boost the BLEU score by about 0.8%. This results from the fact that the lattice include more searching space than N-best list. Secondly, the tuning strategy discussed in the Chapter 6 plays a crucial role for the success of this work, since if the old language model is directly replaced by the composite language model without re-tuning the parameters, the BLEU improvement is no more than 0.3%. Thirdly, the PLSA model contributes more improvement on BLEU compared to the SLM, which is consistent with the PPL performance.

With respect to the running time, since we are testing the performance on distribute LMs, it is expectable that the lattice rescoring speed is slower in the first-pass decoder, in which the n-gram model is directly loaded into the memory. However, we intend to compare the time consumptions between the standard n-gram model and our composite language model both under the distributed version. We observe the following phenomenon: the composite LMs often (about 70% in frequency) require the access of the WORD-PREDICTOR parameters which have been required

		TUNE	TEST
Moses	44 million	21.64	23.48
n-best	CLM_{tn}^{44}	21.82	23.82
	CLM_{all}^{44}	21.99	23.80
lattice	CLM_{tn}^{44} -notune	21.88	23.85
	CLM_{tn}^{44}	22.15	24.00
	CLM_{all}^{44}	**22.49**	**24.26**
Moses	230 million	22.68	24.98
n-best	CLM_{tn}^{230}	22.97	25.30
	CLM_{all}^{230}	23.02	25.35
lattice	CLM_{tn}^{230} -notune	22.99	25.19
	CLM_{tn}^{230}	23.20	25.51
	CLM_{all}^{230}	**23.48**	**25.73**

Table 8.5: BLEU (%) for lattice and N-best rescoring

recently, since in the multi-stack search, we observe that most of partial parses are very similar to each other. Such situation is not observed for n-gram model. To speed up and reduce the network load, we cache the LM parameters which it recently has required. Moreover, Eq. 4.25 show that the topic distribution become stable when d_k is large. When a partial hypothesis generates more than 50 words, the algorithm automatically keep the 10 topics with maximum SEMANTIZER probabilities, and prune the rest. This trick does not harm the performance but remarkably accelerate the rescoring.

For every document (about 23 sentences), the lattice rescoring with n-gram model is about 16 minutes, but the one with composite language models, such as CLM_{all}^{40} and CLM_{all}^{230}, is about 123 minutes, only 7.7 times slower than n-gram.

9

Conclusion and Future Work

We build a composite language model which integrates well-known n-gram, SLM, and PLSA models under the directed MRF paradigm. The composite language model is trained by performing a convergent N-best list approximate EM algorithm and a follow-up EM algorithm. In order to increase our corpus size up to a billion tokens, we propose a client-server architecture to distribute the LM, stored it on a supercomputer. Finally, we develop an A^*-search-based lattice rescoring method in order to integrate our language model in to a phrased-based MT decoder. We have achieved drastic perplexity reductions and obtained significantly better translation quality measured by the BLEU score of translations in the tasks of reranking the N-best list and rescoring the lattices.

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], "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 et al., 2009], but care needs to be taken here. As we explained in Chapter 7, 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 the 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, the future research directions of this work include:

1. The combination of more advanced topic models with SLM and n-gram. PLSA provides a incomplete probabilistic modeling of text, since it provides no probabilistic model at the level of documents. In PLSA, each document is represented as a vector of numbers (the mixing proportions for topics), and there is no generative probabilistic model for these numbers. This leads to several problems: (1) the number of parameters in the model grows linearly with the size of the corpus, which results in serious problems with overfitting, and (2) it is not clear how to assign probability to a document outside of the training set. Therefore, We plan

to integrate more advanced topic language models such as latent Dirichlet allocation (LDA), which a three-level hierarchical Bayesian model, providing an explicit representation of a document. In this case, we need to follow [Blei et al., 2003] to give efficient approximate inference techniques based on variational methods and an EM algorithm for empirical Bayes parameter estimation.

2. Non-parametric Bayesian smoothing. One severe problem in language modeling is the sparse data problem. Various smoothing techniques have been proposed to combat this problem. In this work, we used the linear interpolation technique. However, the best smoothing technique, Kneser-Ney smoothing, only handles explicit counts (integer values). In this project, there are hidden variables, for example, m headwords, h_{-m}^{-1} in WORD-PREDICTOR, $p(w|w_{-n+1}^{-1}h_{-m}^{-1}g)$, of the composite n-gram/m-SLM/PLSA model, resulting in fractional counts during EM iterations. How to smooth fractional counts due to latent variables in Kneser-Ney's sense in a principled way is a long standing open problem [Goodman, 2001]. Teh [Teh and Jordan, 2010; Teh, 2006] described a hierarchical Bayesian model consisting of Pitman-Yor processes [Pitman, 2006; Pitman and Yor, 1997] as a language model and derived estimation formulas for trigrams based on this model that are generalizations of one of the most successful smoothing techniques, interpolated Kneser-Ney smoothing. In this project, we plan to study how to make use of the hierarchical Pitman-Yor process as part of more sophisticated composite language models. Non-parametric Bayesian smoothing for the composite language model will be a hierarchical extension of the Pitman-Yor process. Smoothing fractional counts due to latent variables in Kneser-Ney's sense in a principled way, might be solved.

3. Integration of our composite language models in the parsing-based decoder in Joshua [Li et al., 2009]. Joshua implements the algorithms in Hiero, a hierarchical phrased translation model developed by Chiang [Chiang, 2007], which assumes a probabilistic synchronous context-free grammar (SCFG). We approximate the

language model probability of generating the English hypothesis e, and its probability $p(e)$, by the product of language model probabilities of English word strings explicitly appeared in e. This approximation is also made by Chiang when he inserts n-grams into SCFG, see Eqs. (29-30) in [Chiang, 2007].

4. In [Tomas et al., 2010; Tomas, 2012], recurrent neural network based language models (RNNLM) shows a significant outperformance on perplexity and WER score based on n-gram. They manage to store the past information of arbitrary length by hidden layers of the neural networks. Due to its simplicity and easy understandability, RNNLMs become hot spots of research on language modeling recently. However, they have following weaknesses: (1) RNNLMs wrap the long-range dependencies between words within a "blackbox" of neural networks. How syntactic and semantic information is encoded into this framework is unclear. (2) To our knowledge, no distributed training algorithms have been proposed for a billion-level large scale training sets. Our composite language models have their advantages on these two aspects. Our future study also includes the composite language model combining the RNNLM with syntactic models and topic models and the distributed version of such model in order to train a large scale training corpus.

Appendix A: Examples of translation results on N-best Ranking

In the following, we give examples of "perfect" sentences, or "only semantically correct" sentences, or "only grammatically correct" sentences, where the digit numbers are the sentence number in the N-best list from Hiero, a denotes 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%-40% of the gold business volume, becoming the major consumers of gold business in Hong Kong.

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 .

References

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

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

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

M. Banko and E. Brill. 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, 2001.

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

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

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

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

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

J. Bilmes and K. Kirchhoff. 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, 2003.

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

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

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

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

C. Chelba, J. Schalkwyk, T. Brants, V. Ha, B. Harb, W. Neveitt, C. Parada and P. Xu. Query language modeling for voice search. *Proceedings of the 2010 IEEE Workshop on Spoken Language Technology* (SLT), 127-132, 2010.

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

C. Chelba. Exploiting syntactic structure for natural language modeling. *Ph.D. Dissertation*, The Johns Hopkins University, 2000.

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

S. Chen. Performance prediction for exponential language models. *In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 450-458, 2009.

S. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. *In Proceedings of the 34th annual meeting on Association for Computational Linguistics.*, 310-318, 1996.

Z. Chi. Statistical properties of probabilistic context-free grammars. *In Computational Linguistics*, 25(1):131-160, 1999.

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

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

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

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

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

D. Gildea and T. Hofmann. Topic-based language models using EM. *The 6th European Conference on Speech Communication and Technology* (EUROSPEECH), 2167-2170, 1999

Y. Goodman. A bit of progress in language modeling. *Computer Speech and Language*, 403-434, 2001.

A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8-12, 2009.

T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd Edition, Springer, 2009

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

F. Jelinek. The dawn of statistical ASR and MT. Computational Linguistics. *Computational Linguistics*, 35(4):483-494, 2009.

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

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

F. Jelinek. *Statistical Methods for Speech Recognition*, MIT Press, 1998

F. Jelinek and R. Mercer. Interpolated estimation of Markov source parameters from sparse data. *Pattern Recognition in Practice*, North Holland Publishers, 1980

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

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

S. Khudanpur and J. Wu. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech and Language*, 14(4):355-372, 2000.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M.Federico, N. Bertoldi et al. Moses: Open Source Toolkit for Statistical Machine Translation. *In Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, 177-180, 2007.

P. Koehn. Statistical significance tests for machine translation evaluation, *The 2004 Conference on Empirical Methods in Natural Language Processing* (EMNLP), 388-395, 2004.

P. Koehn, F. Och and D. Marcu. Statistical Phrase-based Translation. *In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology* , 48-54, 2003

K. Lari and S. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35-56, 1990.

S. Lauritzen. *Graphical Models*. Oxford Press, 1996.

Z. Li, C. Callison-Burch, S. Khudanpur, and W. Thornton. Decoding in Joshua: Open source, parsing-based machine translation. *The Prague Bulletin of Mathematical Linguistics (PBML)*, 91, 2009.

W. Macherey, F. Och, I.Thayer and J. Uszkoreit Lattice-based Minimum Error Rate Training for Statistical Machine Translation. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing* , 25-734, 2008

K. Mark, M. Miller and U. Grenander. Constrained stochastic language models, In *Image Models and Their Speech Model Cousins*, 131-137, Springer-Verlag, 1996.

D. McAllester, M. Collins and F. Pereira. Case-factor diagrams for structured probabilistic modeling. *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* (UAI), 382-391, 2004.

N. Nilsson. *Problem Solving Methods in Artificial Intelligence*, NcGraw-Hill, 1971

P. Norvig. Statistical learning as the ultimate agile development tool. *ACM 17th Conference on Information and Knowledge Management (CIKM)*, Industry Event, 2008.

F. Och. Minimum Error Rate Training in Statistical Machine Translation. *In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, 160-167, 2003

F. Och and H. Ney Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 295-302, 2002.

K. Papineni, S. Roukos, T. Ward and W. Zhu. BLEU: a method for automatic evaluation of machine translation. *Proceedings of the 40th annual meeting on association for computational linguistics (ACL)*, 311-318, 2002.

S. Pietra, V. Pietra and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380-393, 1997

J. Pitman. *Combinatorial Stochastic Processes*, Springer-Verlag, 2006.

J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855-900, 1997.

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

R. Rosenfeld, S. Chen and X. Zhu. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. *Computer Speech and Language*, 15(1): 55-73, 2001.

R. Rosenfeld. Incorporating linguistic structure into statistical language models. *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, 358(1769):1311-1324, 2000.

R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10(2):187-228, 1996

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, 3rd Edition. Prentice Hall, 2010.

L. Schwartz, C. Callison-Burch, W. Schuler and S. Wu. Incremental Syntactic Language Models for Phrase-based Translation. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)* , 620-631, 2011

L. Schwartz, C. Callison-Burch, W. Schuler, S. Wu and M. Rochester. Erratum to Incremental Syntactic Language Models for Phrase-based Translation.

A. Stolcke. SRILM-An Extensible Language Modeling Toolkit. *In Proceedings of the international conference on spoken language processing*, 2:901-904, 2002

Y. Teh and M. Jordan. Hierarchical Bayesian nonparametric models with applications. In *Bayesian Nonparametrics: Principles and Practice*, 158-207, Cambridge University Press, 2010.

Y. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. *The 44th Annual Conference on Association of Computational Linguistics (ACL)* , 985-992, 2006.

M. Tomas. Statistical Language Models based on Neural Networks. *PhD thesis, Brno University of Technology*, 2012.

M. Tomas, K. Martin, B. Lukas, C. Jan and K. Sanjeev Recurrent neural network based language model. *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2010.

N. Ueffing, F. Och, and H. Ney. Generation of Word Graphs in Statistical Machine Translation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)* , 156-163, 2002.

V. Vapnik. 1998. *Statistical Learning Theory*, Springer.

A. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm *IEEE Transactions on Information Theory*, 13(2):260-269, 1967

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

K. Wang, C. Thrasher, E. Viegas, X. Li and P. Hsu. 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, 2010.

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

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

S. Wang, D. Schuurmans, F. Peng and Y. Zhao. 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, 2005.

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

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

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

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

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

Y. Zhang, S. Vogel, A. Emami, K. Papineni, J. Sorensen and J. Quinn. Distributed language modeling. *GALE Book*, Chapter 2.5.1, 252-270, 2011.

Y. Zhang. Structured language models for statistical machine translation. *Ph.D. Dissertation*, Carnegie Mellon University, 2008.

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

Publications

M. Tan, T. Xia, S. Wang and B. Zhou. A corpus level MIRA tuning strategy for machine translation. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2013.

M. Tan, T. Xia, L. Guo and S. Wang. Direct optimization of ranking measures for learning to rank Models. The 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2013.

S. Zhai, T. Xia, **M. Tan** and S. Wang. Direct 0-1 Loss Minimization and Margin Maximization with Boosting. The 27th Annual Conference on Neural Information Processing Systems (NIPS), 2013.

S. Zhai, T. Xia, **M. Tan**, S. Wang and P. Zhang. A Robust Semi-supervised Boosting Method Using Linear Programming. The 1st IEEE Global Conference on Signal and Information Processing, 2013.

M. Tan, W. Zhou, L. Zheng and S. Wang. A scalable distributed syntactic and lexical language model, Computational Linguistics Journal (CL), 38(3): 631-671, 2012.

W. Wang, L. Chen, **M. Tan**, S. Wang and A. Sheth. Discovering fine-grained sentiment in suicide notes. Biomedical Informatics Insights, 5 (Suppl. 1): 137-145, 2012.

M. Tan, W. Zhou, L. Zheng and S. Wang, A scalable distributed syntactic, semantic and lexical language model for machine translation. The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL), 201-210, 2011.