Structured Language Models for Statistical Machine Translation

Ying Zhang

CMU-LTI-09-009

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Abstract

Language model plays an important role in statistical machine translation systems. It is the key knowledge source to determine the right word order of the translation. Standard n-gram based language model predicts the next word based on the $n-1$ immediate left context. Increasing the order of n and the size of the training data improves the performance of the LM as shown by the suffix array language model and distributed language model systems. However, such improvements narrow down very fast after n reaches 6. To improve the n-gram language model, we also developed dynamic n-gram language model adaptation and discriminative language model to tackle issues with the standard n-gram language models and observed improvements in the translation qualities.

The fact is that human beings do not reuse long n-grams to create new sentences. Rather, we reuse the structure (grammar) and replace constituents to construct new sentences. Structured language model tries to model the structural information in natural language, especially the long-distance dependencies in a probabilistic framework. However, exploring and using structural information is computationally expensive, as the number of possible structures for a sentence is very large even with the constraint of a grammar. It is difficult to apply parsers on data that is different from the training data of the treebank and parsers are usually hard to scale up.

In this thesis, we propose x-gram language model framework to model the structural information in language and apply this structured language model in statistical machine translation. The x-gram model is a highly lexicalized structural language model. It is a straight-forward extension of the n-gram language model. Trained over the dependency trees of very large corpus, x-gram model captures the structural dependencies among words in a sentence. The probability of a word given its structural context is smoothed using the well-established smoothing techniques developed for n-gram models. Because x-gram language model is simple and robust, it can be easily scaled up to larger data.

This thesis studies both semi-supervised structure and unsupervised structure. In semi-supervised induction, a parser is first trained over human labeled treebanks. This parser is then applied on a much larger and unlabeled corpus. Tree transformation is applied on the initial structure to maximize the likelihood of the tree given the initial structured language model. When the treebank is not available, as is the case for most of the languages, we propose the "dependency model 1" to induce the dependency structure from the plain text for language modeling as unsupervised learning.

The structured language model is applied in the SMT N-best list reranking and evaluated by the structured BLEU metric. Experiments show that the structured language model is a good complement to the n-gram language model and it improves the translation quality especially on the fluency aspect of the translation. This work of modeling the structural information in a statistical framework for large-scale data opens door for future research work on synchronous bilingual dependency grammar induction.

Acknowledgements

First and foremost, I would like to thank my committee chair and thesis advisor Dr. Stephan Vogel, who carried me through tough phases and believed in me. His encouragement and support made this thesis possible. I have learned a great deal from Stephan in the past seven years in research, project management, people skills and sports. I thank Alex Waibel for his insightful advice and for offering me an excellent environment - Interactive Systems Laboratories - to pursue my degree. I greatly appreciate the invaluable advice and inspiring discussions from Noah Smith. I found every meeting with him a step forward. I want to thank Sanjeev Khudanpur for his expertise and insights.

During my years in graduate school, I have benefit a lot from the profound knowledge and expertise of the faculty members in LTI. I want to thank all the faculty members who have advised me and helped me in the past. To Alon Lavie, Ralf Brown, Bob Frederkin, Tanja Schultz, Lori Levin, Jie Yang, Yiming Yang, Jaime Carbonell, Teruko Mitamura, Eric Nyberg, Alan Black, Jamie Callan, Maxine Eskenazi, John Lafferty, Alex Hauptmann, Roni Rosenfeld, Alex Rudnicky and Richard Stern, thank you for everything.

I have been through a lot of system building and project evaluations during my years as a graduate student and frankly I have learned a lot from this experience. Many thanks to my fellow SMT team-mates Alicia Sagae, Ashish Venegopal, Fei Huang, Bing Zhao, Matthias Eck, Matthis Paulik, Sanjika Hewavitharana, Silja Hildebrand, Mohamed Noamany, Andreas Zollmann, Nguyen Bach, ThuyLinh Nguyen,Kay Rottmann, Qin Gao and Ian Lane for their collaboration, discussion and support during TIDES, GALE, STR-DUST and TRANSTAC projects.

Thanks to my colleagues at the InterAct center: Hua Yu, Susi Burger, Qin Jin, Kornel Laskowski, Wilson Tam, Szu-Chen Stan Jou, Roger Hsiao, , Celine Carraux, Anthony D'Auria, Isaac Harris, Lisa Mauti, Freya Fridy, Stephen Valent, Thilo Kohler, Kay Peterson for the wonderful time and the feeling of a big family.

I learned a lot from my fellow students at LTI. They are smart, talented, hard-working and inspiring. I enjoy the time spent with them, no matter it is in classrooms, in the seminar, a chat in the lounge, TG, or during the LTI ski trip at Seven Springs, LTI movie nights or LTI volleyball games. To Satanjeev Banerjee, Ulas Bardak, Vitor Carvalho, Yi Chang, Chih-yu Chao, Paisarn Charoenpornsawat, Datong Chen, Betty Yee Man Cheng, Shay Cohen, Kevin Collins-Thompson, Madhavi Ganapathiraju , Kevin Gimpel, Qin Gao, Kevin Gimpel, Lingyun Gu, Benjamin Ding-Jung Han, Gregory Hanneman, Yi-Fen Huang, David Huggins-Daines, Chun Jin, Rong Jin, Jae Dong Kim, John Korminek, Mohit Kumar, Brian Langner, Guy Lebanon, Fan Li, Frank Lin, Wei-Hao Lin, Lucian Vlad Lita, Yan Liu, Christian Monson, Paul Ogilvie, Vasco Pedro, Erik Peterson, Kathrina Probst, Yanjun Qi, Sharath Rao, Antoine Raux, Luo Si, Kenji Sagae, Arthur Toth, Mengqiu Wang, Wen Wu, Rong Yan, Hui Grace Yang, Jun Yang, Jian Zhang, Le Zhao, Yangbo Zhu: thank you for the wonderful time.

I also want to thank Radha Rao, Linda Hager, Stacey Young, Cheryl Webber, Mary Jo Bensasi, the staff members of LTI who helped me throughout my thesis.

I have benefited a lot from colleagues in the SMT community. Thanks to Franz Och, Chiprian Chelba, Peng Xu, Zhifei Li, Yonggang Deng, Ahmad Emami, Stanley Chen, Yaser Al-onaizan, Kishore Papineni, David Chiang, Kevin Knight, Philip Resnik, Smaranda Muresan, Ryan McDonald, Necip Fazil Ayan, Jing Zheng and Wen Wang for their suggestions and advice during this thesis work.

For Sunny, my parents, my brother Johnny, sister-in-law Marian, sweet niece Allie, sister Jie, brother-in-law Youqing and my lovely nephew Canxin: this thesis is dedicated to you for your unconditional love and support.

The peril of having an acknowledgement is, of course, to have someone left out unintentionally. For all of you still reading and pondering, you have my deepest gratitude.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Building machines that can translate one human language to another has been a dream for scientists even before the first electronic computer was invented. Early approaches in machine translation research tried to mimic the human's translation process according to linguistic theories where we first parse the source sentence to understand the structure of the sentence and then apply the rules to transform the structure of the original sentence and translate the lexicon/phrases to the target language. Intuitively sound, this approach requires intense human effort to engineer grammar rules for parsing the source language and transforming from the source language to the target. More importantly, natural languages are way more complicated than formal languages and linguistic rules can not explain all phenomena in a consistent method.

Statistical machine translation (SMT) systems [Brown et al., 1993] consider translation as a stochastic process. Each word f in a source sentence $\mathbf{f}$ is translated into the target language. For almost all language pairs, there are not only one-to-one correspondences between the source and target words, thus each f can be translated into different target words e with different probabilities. A decoder in the SMT system searches through all possible translation combinations ($\mathbf{e}$) and select the one with the highest probability $\mathbf{e}^*$ as the final translation.

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \mathbf{P}(\mathbf{f}|\mathbf{e}) \tag{1.1}$$

which can be decomposed into:

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \mathbf{P}(\mathbf{e})\mathbf{P}(\mathbf{f}|\mathbf{e}) \tag{1.2}$$

where $P(\mathbf{e})$ is the probability estimated by the language model (LM) as how likely $\mathbf{e}$ is a "good" sentence and $P(\mathbf{f}|\mathbf{e})$ denotes the probability of $\mathbf{f}$ as $\mathbf{e}$'s translation estimated by the translation model (TM).

The original SMT system as described in [Brown et al., 1993] is based on the word-to-word translation model. Recent years have seen the introduction of phrase-to-phrase based translation systems which learn the translation equivalences at the phrasal-level [Och et al., 1999, Marcu and Wong, 2002, Zhang et al., 2003, Koehn et al., 2003, Venugopal et al.]. The so-called "phrases" are not necessarily linguistically motivated. A "phrase" could contain words that belong to multiple linguistic constituents, such as "go to watch the". In this sense, a "phrase" is actually an n-gram. Even though "phrases" may not have correct linguistic meanings, the correspondence between phrases from the source and target languages encapsulates more contextual information than the word-to-word based models and generates much better translations. When evaluated against human-generated reference translations using n-gram based metrics such as BLEU [Papineni et al., 2001], phrase-based SMT systems are close and sometimes even better than human translations. However when human judges read the translations output from SMT systems, the overall impression is that MT output is "understandable" but often not "grammatical". This is a natural outcome of phrase-based SMT systems because we only consider dependencies in language at n-gram level. n-grams are the fundamental unit in the modeling process for all parts in the SMT system: n-gram based translation models, n-gram based language models and even n-gram based evaluation metrics.

1.1 n-gram Language Model and Its Limitation

A language model is a critical component in an SMT system. Probabilities from the LM help the decoder to make decisions on how to translate a source sentence into a target sentence. Borrowing from work done in the Automatic Speech Recognition (ASR) community, most SMT systems use the

n-gram language model where the choice of the next word depends only on the previous $n-1$ words. Introduced more than half a century ago by Shannon [Shannon, 1951], the n-gram language model turns out to be very robust and effective in ASR despite its simplicity. It is surprisingly hard to beat n-gram models trained with a proper smoothing (e.g. modified Kneser-Ney [Chen and Goodman, 1996]) on abundant training data [Goodman, 2001]. However, the n-gram LM faces a challenge in SMT that is not present in ASR: word reordering. In ASR, acoustic signals are transcribed into words in a monotone order [1] whereas the order of words in the source and target language could be drastically different. For example, in English, the sentence *Suzuki uses a computer* has the order subject *(Suzuki)*, verb *(uses)*, and object *(a computer)*. In the corresponding Japanese sentence, the subject comes first, just as in English, but then the object appears, followed finally by the verb: *Suzuki-ga (Suzuki) konpyuuta-o (computer) tukau (use)*.

Although n-gram models are simple and effective for many applications, they have several limitations [Okanohara and Tsujii, 2007]. n-gram LM cannot determine correctness of a sentence independently because the probability depends on the length of the sentence and the global frequencies of each word in it. For example, $P(\mathbf{e}_1) < P(\mathbf{e}_2)$, where $P(\mathbf{e})$ is the probability of a sentence $\mathbf{e}$ given by an n-gram LM, does not always mean that $\mathbf{e}_1$ is more correct, but instead could occur when $\mathbf{e}_1$ is shorter than $\mathbf{e}_2$, or if $\mathbf{e}_1$ has more common words than $\mathbf{e}_2$. Another problem is that n-gram LMs cannot handle overlapping information or non-local information easily, which is important for more accurate sentence classification. For example, an n-gram LM could assign a high probability to a sentence even if it does not have a verb.

On November 13-14, 2006, a workshop entitled “Meeting of the MINDS: Future Directions for Human Language Technology,” sponsored by the U.S. Government’s Disruptive Technology Office (DTO), was held in Chantilly, VA. “MINDS” is an acronym for Machine Translation, Information Retrieval, Natural Language Processing, Data Resources, and Speech Understanding. These 5 areas were each addressed by a number of experienced researchers. The goal of these working groups was to identify and discuss especially promising future research directions, especially those which are un(der)funded.

[1]There are a few exceptions in languages such as Thai where the graphemes can be of different orders as their corresponding phonemes.

As stated in the Machine Translation Working Group's final report [Lavie et al., 2006, 2007],

> "The knowledge resources utilized in today's MT systems are insufficient for effectively discriminating between good translations and bad translations. Consequently, the decoders used in these MT systems are not very effective in identifying and selecting good translations even when these translations are present in the search space. *The most dominant knowledge source in today's decoders is a target language model (LM).* The language models used by most if not all of today's state-of-the-art MT systems are traditional statistical n-gram models. These LMs were originally developed within the speech recognition research community. MT researchers later adopted these LMs for their systems, often "as is." Recent work has shown that statistical trigram LMs are often too weak to effectively distinguish between more fluent grammatical translations and their poor alternatives. Numerous studies, involving a variety of different types of search-based MT systems have demonstrated that the search space explored by the MT system in fact contains translation hypotheses that are of significantly better quality than the ones that are selected by current decoders, but the scoring functions used during decoding are not capable of identifying these good translations. Recently, MT research groups have been moving to longer n-gram statistical LMs, but estimating the probabilities with these LMs requires vast computational resources. Google, for example, uses an immense distributed computer farm to work with 6-gram LMs. These new LM approaches have resulted in small improvements in MT quality, but have not fundamentally solved the problem. There is a dire need for developing novel approaches to language modeling, specific to the unique characteristics of MT, and that can provide significantly improved discrimination between "better" and "worse" translation hypotheses. "

An example discussed in [Chiang, 2005] clearly illustrates this problem (Figure 1-1). To address the word order issue, SMT decoders usually try many different ways to shuffle the target phrases under a certain constraint (e.g. the Inverted Transduction Grammar - ITG constraints [Wu, 1997]) and depend on the distortion model and language model probabilities to find the best word order.

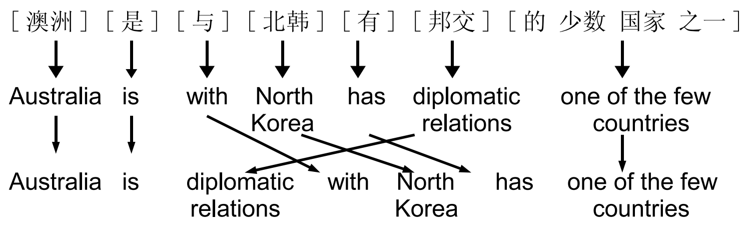

[澳洲] [是] [与] [北韩] [有] [邦交] [的 少数 国家 之一]
Australia is with North Korea has diplomatic relations one of the few countries
Australia is diplomatic relations with North Korea has one of the few countries

Figure 1-1: An example where n-gram LM favors translation with incorrect order.

As the n-gram LM only sees a very local history and has no structure information about the sentence, it usually fails to pick up the hypothesis with the correct word order. In this example, the n-gram LM prefers the translation with the incorrect word order because of the n-grams crossing the phrase boundaries *diplomatic relations with* and *has one of the few* have high probabilities.

1.2 Motivation

It is very unlikely for people to say the same sequence, especially the same long sequence of words again and again. For example, n-gram *Zambian has ordered the British writer Clarke to leave* as occurred in sentence:

> *Zambian interior minister Chikapwasha said that Zambia has ordered the British writer Clarke to leave the country within 24 hours.*

might never occur again in any newspaper after its first appearance.

In reality, we reuse the structure/grammar from what we have learned and replace the "replaceable" parts (constituents) with different contents to form a new sentence, such as:

> *Columbian foreign minister Carolina Barco said that Columbia has ordered the American reporter John Doe to leave the country within 48 hours.*

Structure can be of different generality. A highly generalized grammar could simply state that an English sentence needs to have a subject, a verb and an object in the order of Subject-Verb-Object. On the other hand, a highly lexicalized grammar lists the valid sentence templates by replacing some parts of the sentence with placeholders while keeping the rest word forms. Such templates or rules could look like: X_1 *said that* X_2 *has ordered* X_3 *to leave the country with* X_4 *hours.*

Inspired by the success of the Hiero system [Chiang, 2005] where a highly lexicalized synchronous grammar is learned from the bilingual data in an unsupervised manner by replacing the shorter phrase pairs in a long phrase pair with a generic placeholder to capture the structural mapping between the source and target language, we propose to learn the structure of the language by inducing a highly lexicalized grammar using a unsupervised parsing in which constituents in a sentence are identified hierarchically. With such lexicalized grammar, the language model could capture more structural information in languages and as a result, improve the quality of the statistical machine translation systems.

1.3 Thesis Statement

In this thesis, we propose a general statistical framework to model the dependency structure in natural language. The dependency structure of a corpus comes from a statistical parser and the parsing model is learned through three types of learning: supervised learning from the treebank, semi-supervised learning from initial parsed data and unsupervised learning from plain text. The x-gram structural language model is applied to rerank the N-best list output from a statistical machine translation system. We thoroughly study the impact of using the structural information on the quality of the statistical machine translation.

1.4 Outline

The rest of the document is organized as follows. Chapter 2 reviews the literature of related research on language modeling using the structure information and structure induction. In Chapter 3 we introduce the suffix array language model and distributed language model which push the n-gram language model to its limit by using very large training corpus and long histories. Chapter 4 discusses another way to improve over the generative n-gram language model through discriminative training. Chapter 5 describes the x-gram language model framework which models the structural information in language. Semi-supervised and unsupervised structure induction work is presented in Chapter 6 and we discuss the evaluation methods and the experimental results in Chapter 7. In the end, we propose several future research directions in Chapter 8.

Chapter 2

Related Work

In this chapter, we review some of the major work that is related to this proposal in the area of utilizing structure information in language modeling, using structure information in N-best list reranking and unsupervised structure induction.

2.1 Language Model with Structural Information

It has long been realized that n-gram language models cannot capture the long-distance dependencies in the data. Various alternative language model approaches have been proposed, mainly by the speech recognition community, to incorporate the structural information in language modeling. In speech recognition, the objective is to predict the correct word sequence given the acoustic signals. Structural information could improve the LM so that the prediction becomes more accurate. In SMT, produce correct structure itself is an objective. Translating a source sentence into some target words/phrases and put them together randomly could yield meaningless sentences to a human reader.

2.1.1 Language Model With Long-distance Dependencies

Trigger Models

The trigger language model as described in [Lau et al., 1993] and [Rosenfeld, 1994] considers a *trigger pair* as the basic information bearing element. If a word sequence A is significantly correlated with another word sequence B, then $(A \rightarrow B)$ is considered a "trigger pair", with A being the trigger and B the triggered sequence. When A occurs in the document, it triggers B, causing its probability estimate to change. The trigger model is combined with the n-gram language model in a maximum entropy framework and the feature weights are trained using the generalized iterative scaling algorithm (GIS) [Darroch and Ratcliff, 1972].

Skipped n-gram Language Model

As one moves to larger and larger n-grams, there is less and less chance of having seen the exact context before; but the chance of having seen a similar context, one with most of the words in it, increases. Skipping models [Rosenfeld, 1994, Huang et al., 1993, Ney et al., 1994, Martin et al., 1999, Manhung Siu; Ostendorf, 2000] make use of this observation. In skipping n-gram models, a word conditions on a different context than the previous n-1 words. For instance, instead of computing $P(w_i|w_{i-2}w_{i-1})$, skipped models compute $P(w_i|w_{i-3}w_{i-2})$. For w_i, there are C_{i-1}^{n-1} different skipped $n-1$ gram context. To make the computation feasible, skipped n-gram models only consider those skipped n-grams in a short history context, such as skipped 2-gram, 3-grams occur in the previous 4 words. Because of this limitation, skipped n-gram models do not really address the long distance dependencies in language. Skipping models require both a more complex search and more space and lead to marginal improvements [Goodman, 2001].

LM By Syntactic Parsing

With the recent progress in statistical parsing [Charniak, 1997, Collins, 1997, Klein and Manning, 2003], a straightforward way to use syntactic information in language modeling would be the use of the parser probability for a sentence as the LM probability [Charniak, 2001, Roark, 2001, Lafferty et al., 1992]. Parsers usually require the presence of complete sentences. Most of the LM by parsing work is applied on the N-best list reranking tasks. The left-to-right parser [Chelba and Jelinek, 1998, Chelba, 2000, Xu et al., 2002, 2003] makes it possible to be integrated in the ASR decoder because the syntactic structure is built incrementally while traversing the sentence from left to right. Syntactic structure can also be incorporated with the standard n-gram model and other topic/semantic dependencies in a maximum entropy model [Wu and Khudanpur, 1999].

Here we go over some details of the head-driven parser [Charniak, 2001] to illustrate how typical parsers assign probabilities to a sentence. The probability of a parse is estimated to be:

$$\begin{aligned} P(\pi) &= \prod_{c \in \pi} P(t|l,m,u,i) \\ &\cdot P(h|t,l,m,u,i) \\ &\cdot P(e|l,t,h,m,u) \end{aligned} \tag{2.1}$$

which is the product of probabilities of all the constituents c in the parse π. For each constituent c, the parser first predicts the pre-terminal (POS tag) t tag for the head-word of c, conditioned on the non-terminal label l (e.g., c is NP), the non-terminal label m of c's parent $\hat{c}$, head word m of $\hat{c}$ and the POS tag i of the head word of $\hat{c}$. Then the parser estimates the probability of the head word of c given t, l, m, u and i. In the last step, the parser expand c into further constituents e and estimates this expansion conditioned on l, t, h, m and u. The equation 2.1 is rather complicated. For example, the probability of the prepositional phrase PP in Figure 2-1 is estimated as:

$$\begin{aligned} P(PP) &= P(\text{prep}|\text{PP}, \text{VP}, \text{put}, \text{verb}) \\ &\cdot P(\text{in}|\text{prep}, \text{PP}, \text{VP}, \text{put}, \text{verb}) \\ &\cdot P(\text{prep NP}|\text{in}, \text{prep}, \text{PP}, \text{VP}, \text{put}) \end{aligned} \tag{2.2}$$

[Charniak, 2001] reported significant reduction in perplexity compared with the trigram LM when evaluated on the standard Penn Treebank data which are all grammatical sentences [Marcus et al., 1993]. Unfortunately, the improvement in perplexity reduction has not shown consistent

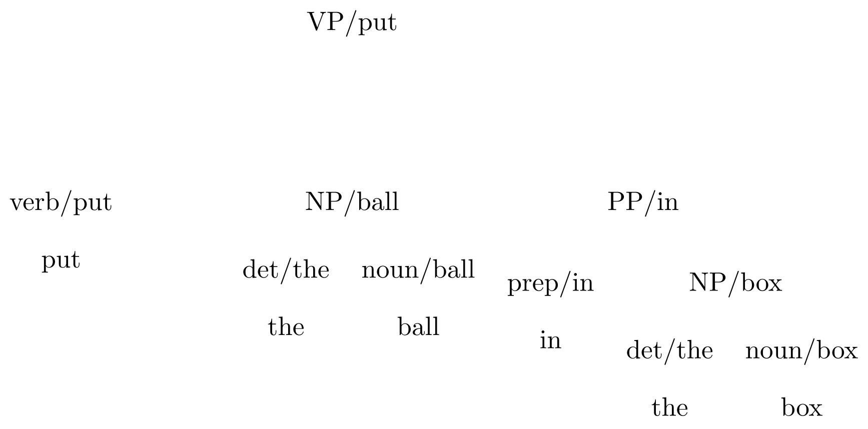


Figure 2-1: A parse tree of *put the ball in the box* with head words.

improvements in machine translation [Charniak et al., 2003]. [Och et al., 2004] even show that the statistical parser (Collins parser) assigns a higher probability to the ungrammatical MT output and lower probabilities to presumably grammatical human reference translations (table 2.1).

Translation	model-best	oracle-best	reference
logProb(parse)	-147.2	-148.5	-154.9

Table 2.1: Average log probabilities assigned by the Collins parser to the model-best, oracle-best and the human generated reference translations.

This counter-intuitive result is due to the fact that the parser is trained to parse the grammatical sentences only and the parsing probabilities are optimized to generate parse trees close to the human parses (treebank) rather than to discriminate grammatical sentences from the ungrammatical ones. In other words, there is a mismatch between the objective of the statistical parser and the structured language model. A parser assigns a high probability to a parse tree that best "explains" a grammatical sentence amongst all possible parses whereas the language model needs to assign a high probability to a hypothesis that is most likely to be grammatical compared with other hypotheses. We therefore consider using the parser probability directly as the LM probability to be ill-motivated.

2.1.2 Factored Language Model

Factored LM approaches in general treat each word in the sentence as a bag of lexical factors such as the surface form, part-of-speech, semantic role, etc. When estimating the probability of a sentence, the LM predicts the next word and its other factors given the factorized history. In factored LM, the structure is flat. The hierarchical syntactic structure is not considered in these approaches.

POS LM

A very simple factored LM is the class-based LM where words and its word-class are used as the factors. The word-class can be learned from the data via various clustering methods, or it can simply be part-of-speech tag of the word.

[Jelinek, 1990] used POS tags as word classes and introduced the POS LM in a conditional probabilistic model where,

$$P(\mathbf{e}) \approx \sum_{\mathbf{t}} \prod_{i} P(e_i|t_i)P(t_i|t_1^{i-1}). \tag{2.3}$$

The conditional POS LM is less effective than the trigram word-based LM because the word prediction solely depends on its tag and the lexical dependency with the previous words are deleted.

The joint probabilistic POS LM presented in [Heeman, 1998] estimates the joint probability of both words and tags:

$$P(\mathbf{e}, \mathbf{t}) = \prod_{i} P(e_i, t_i|e_1^{i-1}, t_1^{i-1}). \tag{2.4}$$

It has been shown that the joint model is superior to the conditional model for POS LM [Johnson, 2001].

SuperTagging LM

Supertags are the elementary structures of Lexicalized Tree Adjoining Grammars (LTAGS) [Joshi et al., 1975]. Supertagging [Bangalore and Joshi, 1999] is similar to part-of-speech (POS) tagging. Besides POS tags, supertags contain richer linguistic information that impose complex constraints in a local context. Each supertag is lexicalized, i.e., associated with at least one lexical item. Usually a lexical item has many supertags, each represents one of its linguistic functionalities given a certain context. Assigning the proper supertag for each word which satisfies the constraints between these supertags is a process of supertagging which is also called "almost parsing" [Bangalore, 1996]. With the training data and testing data all tagged by the supertags, the probability of the testing sentence can be calculated as the standard class-based LM (equation 2.3). The SuperARV Language Model [Wang and Harper, 2002] is based on the same idea except that its grammar formalism is different[1]. The SuperARV LM works similar to other joint-probability class-based LMs by estimating the joint probability of words and tags:

$$\begin{aligned} P(\mathbf{e}, \mathbf{t}) &= \prod_i P(e_i t_i | w_1^{i-1} t_1^{i-1}) \\ &= \prod_i P(t_i | w_1^{i-1} t_1^{i-1}) \cdot P(w_i | w_1^{i-1} t_1^{i}) \\ &\approx P(t_i | w_{i-2}^{i-1} t_{i-2}^{i-1}) \cdot P(w_i | w_{i-2}^{i-1} t_{i-2}^{i}) \end{aligned} \tag{2.5}$$

It is clear from this equation that the SuperARV LM does not consider the structure of the sentence, it is a class-based LM where the class is fine-grained and linguistic driven.

2.1.3 Bilingual Syntax Features in SMT Reranking

[Och et al., 2004] experimented with hundreds of syntactic feature functions in a log-linear model to discriminative rerank the N-best list computed with a then state-of-the-art SMT system. Most of the features used in this work are syntactically motivated and consider the alignment information between the source sentence and the target translation. Only the non-syntactic IBM model 1 feature $P(\mathbf{f}|\mathbf{e})$ improves the BLEU score from the baseline 31.6 to 32.5 and all other bilingual syntactic features such as the tree-to-string Markov fragments, TAG conditional bigrams, tree-to-tree and tree-to-string alignment features give almost no improvement.

2.1.4 Monolingual Syntax Features in Reranking

Similar to [Och et al., 2004], [Hasan et al., 2006] reported experiments using syntactically motivated features to a statistical machine translation system in a reranking framework for three language

[1]The SuperARV LM is based on the Constraint Dependency Grammar (CDG) [Harper and Helzerman, 1995]

pairs (Chinese-English, Japanese-English and Arabic-English).

Three types of syntactic features are used in addition to the original features from the SMT system: supertagging [Bangalore and Joshi, 1999] with lightweight dependency analysis (LDA) [Bangalore, 2000], link grammar [Sleator and Temperley, 1993], and a maximum entropy based chunk parser [Bender et al., 2003]. Using the log-likelihood from the supertagger directly does not have a significant improvement. Link grammar is used as a binary feature for each sentence, 1 for a sentence that could be parsed by the link grammar and 0 if the parsing fails. The link grammar feature alone does not improve the performance, while combined with the supertagging/LDA result in about 0.4∼1.3 improvement of BLEU score. The MaxEnt chunker determines the corresponding chunk tag for each word of an input sequence. An n-gram model is trained on the WSJ corpus where chunks are replaced by chunk tags (total 11 chunk types). This chunk tag n-gram model is then used to rescore the chunked N-best list. The chunking model gives comparable improvement as the combination of supertagging/LDA+Link grammar. Combining three models together, the reranking achieves an overall improvement of 0.7, 0.5 and 0.3 in BLEU score for Chinese-English, Japanese-English and Arabic-English testing data.

In [Collins et al., 2005] the reranking model makes use of syntactic features in a discriminative training framework. Each hypothesis from ASR is parsed by the Collins parser and information from the parse tree is incorporated in the discriminative model as a feature vector $\vec{\Phi}$. These features are much more fine-grained than the binary feature (parsable or not parsable) used in [Hasan et al., 2006]. For example, one feature function in $\vec{\Phi}$ might be the count of CFG rules used during parsing and another feature be the bigram lexical dependencies within the parse tree. Each feature function is associated with a weight. The perceptron learning [Collins, 2002] is used to train the weight vector. Tested on the Switchboard data, the discriminatively trained n-gram model reduced the WER by 0.9% absolute. With additional syntactic features, the discriminative model give another 0.3% reduction.

2.2 Unsupervised Structure Induction

Most of the grammar induction research are supervised, i.e., learning the grammar from an annotated corpus, namely treebanks using either generative models [Collins, 1997] or discriminative models [Charniak, 1997, Turian and Melamed, 2005, 2006a,b, Turian et al., 2006].

Treebanks exist for only a small number of languages and usually cover very limited domains. To statistically induce hierarchical structure over plain text has received a great deal of attention lately [Clark, 2001, Klein and Manning, 2002, 2001, Klein, 2005, Magerman and Marcus, 1990, Paskin, 2001, Pereira and Schabes, 1992, Redington et al., 1998, Stolcke and Omohundro, 1994, Wolff, 1988, Drábek and Zhou, 2000]. It is appealing even for high-density languages such as English and Chinese. There are four major motivations in the unsupervised structure induction:

- To show that patterns of langauge can be learned through positive evidence. In linguistic theory, linguistic nativism claims that there are patterns in all natural languages that cannot be learned by children using positive evidence alone. This so-called *porterty of the stilulus* (POTS) leads to the conclusion that human beings must have some kind of innate linguistic capability [Chomsky, 1980]. [Clark, 2001] presented various unsupervised statistical learning algorithms for language acquisition and provided evidence to show that the argument from the POTS is unsupported;
- To bootstrap the structure in the corpus for large treebank construction [Zaanen, 2000];
- To build parsers from cheap data [Klein and Manning, 2004, Yuret, 1998, Carroll and Charniak, 1992, Paskin, 2001, Smith and Eisner, 2005].
- To use structure information for better language modeling [Baker, 1979, Chen, 1995, Ries et al., 1995].
- To classify data (e.g. enzymes in bioinformatics research) based on the structural information discovered from the raw data [Solan, 2006].

Our proposed work is most related to the fourth category.

2.2.1 Probabilistic Approach

[Klein and Manning, 2001] developed a generative constituent context model (CCM) for the unsupervised constituency structure induction. Based on the assumption that constituents appear in constituent contexts, CCM transfers the constituency of a sequence to its containing context and pressure new sequences that occur in the same context into being parsed as constituents in the next round. [Gao and Suzuki, 2003] and [Klein and Manning, 2004] show that dependency structure can also be learnt in an unsupervised manner. Model estimation is crucial to the probabilistic approach, [Smith and Eisner, 2005] proposed the contrastive estimation (CE) to maximize the probability of the training data given an artificial "neighborhood". CE is more efficient and accurate than the EM algorithm and yields the state-of-the-art parsing accuracy on the standard test set.

Discriminative methods are also used in structure induction. Constituent parsing by classification [Turian and Melamed, 2005, 2006a, Turian et al., 2006] uses variety types of features to classify a span in the sentence into one of the 26 Penn Treebank constituent classes.

2.2.2 Non-model-based Structure Induction

Several approaches attempt to learn the structure of the natural language without explicit generative models.

One of the earliest methods for unsupervised grammar learning is that of [Solomonoff, 1959]. Solomonoff proposed an algorithm to find repeating patterns in strings. Though extremely inefficient, the idea of considering patterns in strings as nonterminals in a grammar has influenced many papers in grammar induction [Knobe and Yuval, 1976, Tanatsugu, 1987].

[Chen, 1995] followed the Solomonoff's Bayesian induction framework [Solomonoff, 1964a,b] to find a grammar G^* with the maximum *a posterior* probability given the training data $\mathcal{E}$. The grammar is initialized with a trivial grammar to cover the training data $\mathcal{E}$. At each step, the induction algorithm tries to find a modification to the current grammar (adding additional grammar rules) such that the posterior probability of the grammar given $\mathcal{E}$ increases.

[Magerman and Marcus, 1990] introduced an information-theoretic measure called *generalized mutual information* (GMI) and used the GMI to parse the sentence. The assumption is that constituent boundaries can be detected by analyzing the mutual information values of the POS n-grams within the sentence. [Drábek and Zhou, 2000] uses a learned classifier to parse Chinese sentences bottom-up. The classifier uses information from POS sequence and measures of word association derived from co-occurrence statistics.

[Yuret, 1998] proposes *lexical attraction model* to represent long distance relations between words. This model links word pairs with high mutual information greedily and imposes the structure on the input sentence. Reminiscent of this work is [Paskin, 2001]'s *grammatical bigrams* where the syntactic relationships between pairs of words are modeled and trained using the EM algorithm.

[Zaanen, 2000] presented Alignment-Based Learning (ABL) inspired by the string edit distance to learn the grammar from the data. ABL takes unlabeled data as input and compares sentence pairs in the data that have some words in common. The algorithm finds the common and different parts between the two sentences and use this information to find interchangeable constituents.

Chapter 3

Very Large LM

The goal of a language model is to determine the probability, or in general the “nativeness” of a word sequence $\mathbf{e}$ given some training data.

Standard n-gram language models collect information from the training corpus and calculate n-gram statistics offline. As the corpus size increases, the total number of n-gram types increases very fast (Table 3.1 and Table 3.2). Building a high order n-gram language model offline becomes very expensive in both time and space [Goodman, 2001].

In this chapter, we describe techniques to build language model using very large training corpus and utilizing higher order n-grams. Suffix array language model (SALM) allows one to use arbitrarily long history to estimate the language model probability of a word. Distributed language model makes it possible to use arbitrarily large training corpus. Database language model (DBLM) is used when we are provided with n-gram frequencies from the training corpus and the original corpus is not available.

n	Types	Tokens
1	33,554	4,646,656
2	806,201	4,518,690
3	2,277,682	4,390,724
4	3,119,107	4,262,867
5	3,447,066	4,135,067
6	3,546,095	4,008,201

Table 3.1: Number of n-gram types in FBIS data (4.6M words).

n	Type	Token
1	1,607,516	612,028,815
2	23,449,845	595,550,700
3	105,752,368	579,072,585
4	221,359,119	562,595,837
5	324,990,454	562,598,343
6	387,342,304	562,604,106
7	422,068,382	562,612,205
8	442,459,893	562,632,078
9	455,861,099	562,664,071
10	465,694,116	562,718,797

Table 3.2: Number of n-gram types in a corpus of 600M words.

3.1 Suffix Array Language Model

3.1.1 Suffix Array Indexing

Suffix array was introduced as an efficient method to find instances of a string in a large text corpus. It has been successfully applied in many natural language processing areas [Yamamoto and Church, 2001, Ando and Lee, 2003].

For a monolingual text $\mathcal{A}$ with N words, represent it as a stream of words: $a_1a_2\dots a_N$. Denote the suffix of $\mathcal{A}$ that starts at position i as $A_i = a_ia_{i+1}\dots a_N$. $\mathcal{A}$ has N suffixes: $\{A_1, A_2, \dots, A_N\}$. Sort these suffixes according to their lexicographical order and we will have a sorted list, such as $[A_{452}, A_{30}, A_1, A_{1511}, \dots, A_7]$. Create a new array X with N elements to record the sorted order, for example, $X = [452, 30, 1, 1511, \dots, 7]$. We call X the *suffix array* of corpus $\mathcal{A}$.

Formally, suffix array is a sorted array X of the N suffixes of $\mathcal{A}$, where $X[k]$ is the starting position of the k-th smallest suffix in $\mathcal{A}$. In other words, $A_{X[1]} < A_{X[2]} < \ldots < A_{X[N]}$, where "<" denotes the lexicographical order. Figure 3-1 illustrates the procedure of building the suffix array from a simple corpus.

Corpus $\mathcal{A}$: $a_1, a_2, \ldots, a_N$

Word Position	1	2	3	4	5	6	7	8	9	10
Word a_{pos}	*how*	*do*	*you*	*say*	*how*	*do*	*you*	do	*in*	*chinese*

Suffixes:

A_1:	*how do you say how do you do in chinese*
A_2:	*do you say how do you do in chinese*
A_3:	*you say how do you do in chinese*
A_4:	*say how do you do in chinese*
A_5:	*how do you do in chinese*
A_6:	*do you do in chinese*
A_7:	*you do in chinese*
A_8:	*do in chinese*
A_9:	*in chinese*
A_{10}:	*chinese*

Sorting all the suffixes:

A_{10}:	*chinese*
A_8:	*do in chinese*
A_6:	*do you do in chinese*
A_2:	*do you say how do you do in chinese*
A_5:	*how do you do in chinese*
A_1:	*how do you say how do you do in chinese*
A_9:	*in chinese*
A_4:	*say how do you do in chinese*
A_7:	*you do in chinese*
A_3:	*you say how do you do in chinese*

Suffix Array X:

Index: k	1	2	3	4	5	6	7	8	9	10
$X[k]$	10	8	6	2	5	1	9	4	7	3

Figure 3-1: Indexing the corpus using the Suffix Array

The sorting of set $\{A_1, A_2, \ldots, A_N\}$ can be done in $\log_2(N+1)$ stages and requires $O(N \log N)$ time in the worse case [Manber and Myers, 1993]. The fast sorting algorithm requires additional data structure to keep track of the partially sorted suffixes and thus requires additional memory in indexing. In most of our applications, suffix array only needs to be built once and speed is not a major concern. We would rather use a $O(N \log N)$ algorithm which is relatively slower but could index a much larger corpus given the limited amount of memory.

In our implementation, each word a_i is represented by a 4-byte $vocId$ and each suffix is a 4-byte pointer pointing to the starting position in $\mathcal{A}$. Thus $8N$ bytes memory are needed in order to index the corpus $\mathcal{A}$.

3.1.2 Estimating n-gram Frequency

With the suffix array built, we can access the corpus and estimate the frequency of any n-gram's occurrence in the data. For an n-gram $\tilde{e}$, we run two binary search on the sorted suffix array to locate the range of $[L, R]$ where this n-gram occurs. Formally, $L = \arg\min_i A_{X[i]} \geq \tilde{e}$ and $R = \arg\max_i A_{X[i]} \leq \tilde{e}$. The frequency of $\tilde{e}$'s occurrence in the corpus is then $R - L + 1$. The binary search requires $O(logN)$ steps of string comparisons and each string comparison contains $O(n)$ word/character comparisons. Overall, the time complexity of estimating one n-gram's frequency is $O(nlogN)$.

In the case when the complete sentence is available, estimating frequencies of all embedded n-grams in a sentence of m words takes:

$$\sum_{n=1}^{m}(m-n+1).nlogN = \frac{m^3+3m^2+2m}{6}logN, \tag{3.1}$$

which is $O(m^3 logN)$ in time. [Zhang and Vogel, 2005] introduced a search algorithm which locates all the $m(m+1)/2$ embedded n-grams in $O(m \cdot log\mathcal{N})$ time. The key idea behind this algorithm is that the occurrence range of a long n-gram has to be a subset of the occurrence range of its shorter prefix. If we start with locating shorter n-grams in the corpus first, then resulting range $[L, R]$ can be used as the starting point to search for longer n-grams.

Figure 3-2 shows the frequencies of all the embedded n-grams in sentence "*since 2001 after the incident of the terrorist attacks on the united states*" matched against a 26 million words corpus. For example, unigram "*after*" occurs 4.43×10^4 times, trigram "*after the incident*" occurs 106

n	since	2001	after	the	incident	of	the	terrorist	attacks	on	the	united	states
1	2.19×10^4	7559	4.43×10^4	1.67×10^6	2989	6.9×10^5	1.67×10^6	6160	9278	2.7×10^5	1.67×10^6	5.1×10^4	3.78×10^4
2		165	105	1.19×10^4	1892	34	2.07×10^5	807	1398	1656	5.64×10^4	3.72×10^4	3.29×10^4
3			6	56	106	6	3	162	181	216	545	605	2.58×10^4
4				0	0	0	1	0	35	67	111	239	424
5					0	0	0	0	0	15	34	77	232
6						0	0	0	0	0	10	23	76
7							0	0	0	0	0	7	23
8								0	0	0	0	0	7

Figure 3-2: Frequencies of all the embedded n-grams in sentence "*since 2001 after the incident of the terrorist attacks on the united states.*"

times. The longest n-gram that can be matched is 8-gram "*of the terrorist attacks on the united states*" which occurs 7 times in the corpus. Given the n-gram frequencies, we can estimate different language model statistics of this sentence.

3.1.3 Nativeness of Complete Sentences

We introduce the concept of *nativeness* Q to quantify how likely a sentence $\mathbf{e}$ is generated by a native speaker. A correct sentence should have higher nativeness score than an ill-formatted one. Unlike the sentence *likelihood*, which is defined as the probability of this sentence generated by a language model, nativeness is only a score of real value and does not need to be probability. Since we can not ask a native speaker to assign scores to a sentence for its nativeness, we need to estimate this value based on statistics calculated from a collection of sentences $\mathcal{E}$ which resembles what a native speaker would generate.

Before we describe how nativeness is estimated, we first introduce the related notation used in the following discussions. An English[1] sentence $\mathbf{e}$ of length J is a sequence of J words: $e_1, \ldots, e_i, \ldots, e_j, \ldots, e_J$, or e_1^J for short. e_i^j denotes an n-gram $e_i e_{i+1} \ldots e_j$ embedded in the sentence. We use $\tilde{e}$ to represent a generic n-gram when n is unspecified. $C(\tilde{e}|\mathcal{E})$ is the frequency of $\tilde{e}$ in corpus $\mathcal{E}$ and $Q(\mathbf{e}|\mathcal{E})$ denotes the nativeness of $\mathbf{e}$ estimated based on $\mathcal{E}$. When there is only one corpus, or the identity of $\mathcal{E}$ is clear from the context, we simply use $C(\tilde{e})$ and $Q(\mathbf{e})$ instead of their full form.

We propose 4 methods to estimate $Q(\mathbf{e})$ from the data.

- Q_c: Number of n-grams matched.

[1]English is used here for the convenience of description. Most techniques developed in this proposal are intended to be language-independent.

The simplest metric for sentence nativeness is to count how many n-grams in this sentence can be found in the corpus.

$$Q_c(e_1^J) = \sum_{i=1}^{J} \sum_{j=i}^{J} \delta(e_i^j) \tag{3.2}$$

$$\delta(e_i^j) = \begin{cases} 1 & : \quad C(e_i^j) > 0 \\ 0 & : \quad C(e_i^j) = 0 \end{cases} \tag{3.3}$$

For example, Q_c for sentence in Figure 3-2 is 52 because 52 n-grams have non-zero counts.

- $Q_{l(n)}$: Average interpolated n-gram conditional probability.

$$Q_{l(n)}(e_1^J) = \left(\prod_{i=1}^{J} \sum_{k=1}^{n} \lambda_k P(e_i | e_{i-k+1}^{i-1}) \right)^{\frac{1}{J}} \tag{3.4}$$

$$P(e_i | e_{i-k+1}^{i-1}) \approx \frac{C(e_{i-k+1}^{i})}{C(e_{i-k+1}^{i-1})} \tag{3.5}$$

$P(e_i | e_{i-k+1}^{i-1})$ is the maximum-likelihood estimation based on the frequency of n-grams. λ_k is the weight for the k-gram conditional probability, $\sum \lambda_k = 1$. λ_k can be optimized by using the held-out data or simply use some heuristics to favor long n-grams.

$Q_{l(n)}$ is similar to the standard n-gram LM except that the probability is averaged over the length of the sentence. This is to prevent shorter sentences being unfairly favored.

- Q_{nc}: Sum of n-grams' non-compositionality. Unlike Q_c where all the matched n-grams are equally weighted, Q_{nc} weights the n-gram by their *non-compositionality* [Zhang et al., 2006]. *Non-compositionality* tries to measure how likely

  > a sequence of two or more consecutive words, that has characteristics of a syntactic and semantic unit, and whose exact and unambiguous meaning or connotation cannot be derived directly from the meaning or connotation of its components [Choueka, 1988].

For each n-gram $\tilde{e}$, the null hypothesis states that $\tilde{e}$ is composed from two unrelated units and *non-compositionality* measures how likely this hypothesis is not true.

We test the null hypothesis by considering all the possibilities to cut/decompose it into two short n-grams and measure the collocations between them. For example $\tilde{e}$ "*the terrorist attacks on the united states*" could be decomposed into ("*the*", "*terrorist attacks on the united states*") or ("*the terrorist*", "*attacks on the united states*"), ..., or ("*the terrorist attacks on the united*", "*states*"), in all n-1 different ways. For each decomposition, we calculate the point-wise mutual information (PMI) [Fano, 1961] between the two short n-grams. The one with the minimal PMI is the most "natural cut" for this n-gram.

The PMI over the natural cut quantifies the *non-compositionality* I_{nc} of an n-gram $\tilde{e}$. The higher the value of $I_{nc}(\tilde{e})$, the less likely $\tilde{e}$ is composed from two short n-grams just by chance. In other words, $\tilde{e}$ is more likely to be a meaningful constituent [Yamamoto and Church, 2001].

Define Q_{nc} formally as:

$$Q_{nc}(e_1^J) = \sum_{i=1}^{J} \sum_{j=i+1}^{J} I_{nc}(e_i^j) \tag{3.6}$$

$$I_{nc}(e_i^j) = \begin{cases} \min_k I(e_i^k; e_{k+1}^j) & : C(e_i^j) > 0 \\ 0 & : C(e_i^j) = 0 \end{cases} \tag{3.7}$$

$$I(e_i^k; e_{k+1}^j) = log \frac{P(e_i^j)}{P(e_i^k)P(e_{k+1}^j)} \tag{3.8}$$

The n-gram probabilities in equation 3.8 are estimated by the maximum-likelihood estimation.

- Q_t: Sum of pointwise mutual information of the distant n-gram pairs

  Q_{nc} calculates the PMI of two adjacent n-grams and uses the sum to measure the *non-compositionality* of the sentence. Q_t calculates the PMI of any non-adjacent n-gram pairs and uses the sum to measure the *coherence* inside a sentence. This is inspired by the single-word trigger model developed in [Rosenfeld, 1994]. We extend the concept from word-triggers to phrase-triggers in this thesis. Instead of considering each trigger as a feature and combining all the features in a log-linear model, we sum up PMI values of all triggers.

Define Q_t as:

$$Q_t(e_1^J) = \sum_{i_1=1}^{J-g-1} \sum_{j_1=i_1}^{J-g-1} \sum_{i_2=j_1+g+1}^{J} \sum_{j_2=i_2}^{J} I(e_{i_1}^{j_1}; e_{i_2}^{j_2}) \tag{3.9}$$

g is the minimum "gap" length between two n-grams.

3.1.4 Smoothing

Smoothing is one of the most important issue in language modeling. The key idea of smoothing is to discount some probability mass of observed events so that there is a little bit of probability mass left for unseen events. There are many smoothing techniques in the language modeling literature, and so far the modified Kneser-Ney smoothing has been shown to be most effective [Chen and Goodman, 1996, Goodman, 2001, James, 2000].

Three nativeness metrics proposed in section 3.1.3 ($Q_{l(n)}$, Q_{nc}, and Q_t) are based on the maximum likelihood estimation of n-gram probabilities. $Q_{l(n)}$ interpolates the conditional probabilities using different history length and thus has some smoothing. Other metrics do not have explicit smoothing built in.

We suspect that smoothing may not be so important for the N-best list reranking task as compared to the LM used in the ASR and SMT decoder. If an n-gram is unseen in the training data, assigning 0 as its probability is not acceptable. In a generative probabilistic LM all n-grams have some positive probabilities, no matter how bizarre the n-gram is. For us, assigning 0 as the nativeness score to an n-gram is fine because the score is only used to discriminate two sentences. To test the impact of smoothing on N-best list reranking, we implemented the Good-Turing smoothing [Good, 1953] in the $Q_{l(n)}$ metric. Good-Turing (GT) smoothing is easy to implement and the count-of-count information can be obtained on-the-fly by scanning the suffix-array indexed corpus [Zhang, 2006], which fits our online model set up well. The implementation was validated on a SMT decoder and showed some small improvements over the MLE estimation. However, when applied on the N-best list re-ranking task, GT-smoothing performed slightly worse than using MLE only.

Model-best	31.44	9.00
$Q_{l(4)}$, equally weighted interpolation	32.16	9.15
$Q_{l(6)}$, equally weighted interpolation	32.41	9.17
$Q_{l(6)}$, favor long n-grams, $\lambda_k = k/21$	32.40	9.18
$Q_{l(6)}$, equally weighted + GT	32.35	9.16

Table 3.3: Comparing different smoothing approaches in N-best list reranking.

Table 3.3 shows the reranking results using different smoothing methods. The N-best list is a $10K$-list generated by Hiero on the Chinese MT03 data. Here we used only a 270M-word corpus for the re-ranking experiment.

The numbers in Table 3.3 are all quite close to each other. We can not conclude that smoothing is not useful in LM for N-best list reranking. We will investigate more in this direction and implement more advanced smoothing methods for better understanding of this issue.

3.2 Distributed LM

We use 4 bytes to represent the vocID of a word in $\mathcal{E}$ and 4 bytes for each suffix array pointer. For a corpus with $\mathcal{N}$ words, the corpus and its suffix array index need $8\mathcal{N}$ bytes[2] to be loaded into the memory. For example, if the corpus has 50 million words, 400MB memory is required. For the English[3] GigaWord[4] corpus which has about 4 billion words, the total memory required is 32GB. It is currently impossible to fit such data into the memory of a single machine in our computing environment[5]. To make use of the large amount of data, we developed a distributed client/server architecture for language modeling. Facilitated with the distributed suffix array indexing of arbitrarily large corpus, the *nativeness* of a sentence given a corpus is calculated on the fly without the pre-calculated n-gram statistics.

[2] $9\mathcal{N}$ bytes if we also index the offset position of a word in the sentence.

[3] Though we used English data for our experiments in this proposal, the approach described here is language independent because all the words in the corpus are mapped into an integer vocID no matter what their surface form and encodings are.

[4] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp? catalogId=LDC2005T12

[5] Our largest machine has only 24GB RAM so far.

3.2.1 Client/Server Paradigm

Client/server is a common paradigm of distributed computing [Leopold, 2001]. The paradigm describes an asymmetric relationship between two types of processes, of which one is the client, and the other is the server. The server process manages some resources (e.g. database, webpages, printers) and offers a service (e.g. database query, returning web page, printing) which can be used the client processes. The client is a process that needs the service in order to accomplish its task. It sends a request (e.g. a query) to the server and asks for the execution of a task (e.g. return all the records that satisfy the query) that is covered by the service.

We split the large corpus $\mathcal{E}$ into $\mathbf{d}$ non-overlapping chunks, $\mathcal{E}_1$,…,$\mathcal{E}_d$, …, $\mathcal{E}_{\mathbf{d}}$. One can easily verify that for any n-gram $\tilde{e}$ the count of its occurrences in $\mathcal{E}$ is the sum of its occurrences in all the chunks, i.e.,

$$C(\tilde{e}|\mathcal{E}) = \sum_{d=1}^{\mathbf{d}} C(\tilde{e}|\mathcal{E}_d) \tag{3.10}$$

where $C(\tilde{e}|\mathcal{E}_d)$ is the frequency of n-gram $\tilde{e}$ in corpus chunk $\mathcal{E}_d$.

Each server [6] loads one chunk of the corpus with its suffix array index. The memory overhead for a server to communicate with the client is very small. Depending on the configuration, each server uses about 1 to 4MB to buffer the communication. The client sends an English sentence $e_1 \ldots e_m$ to each of the servers and requests for the count information of all the n-grams in the sentence. The client collects the count information from all the servers, sums up the counts for each n-gram and then calculates the likelihood of the sentence.

The client communicates with the servers via TCP/IP sockets. In one experiment, we used 150 servers running on 26 computers to serve one client. Multiple clients can be served at the same time if needed. The process of collecting counts and calculating the sentence probabilities takes about 1 to 2 ms for each English sentence (average length 23.5 words). With this architecture, we can easily make use of larger corpora by adding additional data servers. In this experiment, we used all the 2.7 billion word data in the English Gigaword corpus without any technical difficulties. In another configuration, we used a larger corpus of 4 billion words (including LDC Gigaword corpus

[6] Here, a server is a special program that provides services to client processes. It runs on a physical computer but the concept of server should not be confused with the actual machine that runs it.

$\mathbf{f}$: 自从 2001 年 美国 遭受 恐怖 攻击 的 事件 之后
Ref: Since the terrorist attacks on the United States in 2001
$\mathbf{e}^{(1)}$: since 200 year , the united states after the terrorist attacks in the incident
$\mathbf{e}^{(2)}$: since 2001 after the incident of the terrorist attacks on the united states
$\mathbf{e}^{(3)}$: since the united states 2001 threats of terrorist attacks after the incident
$\mathbf{e}^{(4)}$: since 2001 the terrorist attacks after the incident
$\mathbf{e}^{(5)}$: since 200 year , the united states after the terrorist attacks in the incident

Figure 3-3: An example of N-best list.

and BBN web collection[7]) and split it into 89 chunks. Each chunk has about 50M words. We will describe experiments using these two configurations.

3.2.2 N-best list re-ranking

When translating a source language sentence $\mathbf{f}$ into English, the SMT decoder first builds a translation lattice over the source words by applying the translation model and then explores the lattice and searches for an optimal path as the best translation. The decoder uses different models, such as the translation model, n-gram language model, fertility model, and combines multiple model scores to calculate the objective function value which favors one translation hypothesis over the other Och et al. [2004].

Instead of outputting the top hypothesis $\mathbf{e}^{(1)}$ based on the decoder model, the decoder can output N (usually $N = 1000$) alternative hypotheses $\{\mathbf{e}^{(r)}|r = 1, \ldots, N\}$ for one source sentence and rank them according to their model scores.

Figure 3-3 shows an example of the output from a SMT system. In this example, alternative hypothesis $\mathbf{e}^{(2)}$ is a better translations than $\mathbf{e}^{(1)}$ according to the reference (Ref) although its model score is lower.

SMT models are not perfect, it is unavoidable to output a sub-optimal translation as the model-best by the decoder. The objective of N-best list re-ranking is to re-rank the translation hypotheses using features which are not used during decoding so that better translations can emerge as "op-

[7]News stories downloaded by BBN from multiple news websites.

	BLEU	NIST
Model-best	31.92	8.22
Oracle-best	45.04	9.31
Random re-rank	27.91 [27.02, 28.81]	7.89 [7.81, 7.98]

Table 3.4: Randomly re-rank the N-best list.

timal" translations. Empirical experiments have shown that the oracle-best translation from a typical N-best list could be 6 to 10 BLEU points better than the model-best translation. We have to use meaningful features to re-rank the N-best list to improve over the model-best translations. Table 3.4 shows an example of randomly rerank the N-best list. The model-best translation has BLEU score 31.92. The oracle-best translation from this 1000-best list has BLEU score 45.04, 14 BLEU points better than the model-best translation. However, if we just randomly select a hypothesis for each of the testing sentences, the resulting translations are much worse than the model-best. From 5,000 trials, the mean BLEU score is 27.91 with 95% confidence interval at [27.02, 28.81]. This experiment indicates that there are better translations in the N-best list compared to the model-best, but we need to use meaningful features to re-rank them. Through out this thesis, we present different language model features and use them to re-rank the N-best list.

3.2.3 "More data is better data" or "Relevant data is better data"?

Although statistical systems usually improve with more data, performance can decrease if additional data does not resemble the test data. The former claim stats that throwing more data into the training, in this case, language model training should always improves, or at least not hurt the system performances, whereas the latter claims that using only portion of the data which is most relevant to the testing data should outperform using all data together. For N-best list re-ranking, the question becomes: "should we use all data to re-rank the hypotheses for one source sentence, or select some corpus chunks that are believed to be relevant to this sentence?" As the whole data set is chunked into multiple pieces in the distributed LM system and we can either combine statistics from all data chunks (and thus "more data") or choose only a few corpus chunks which we believe are more relevant to the testing data to calculate LM probabilities.

Oracle Study

To answer this question, we designed the following oracle experiment. The English data of 4 billion words is split into 89 chunks. The original data is organized by news source and time. The split keeps this natural order such that all the documents in one chunk come from the same news source and around the same period of time. This is a natural way of clustering the documents without looking into the content. It is a reasonable assumption that "Yemen News" has more middle-east related data than "Seattle Times" and "BBC2005-2006" has more up-to-date coverage than "BBC1999-2000".

We use each individual corpus chunk $\mathcal{E}_d$ to re-rank the N-best list and for each source sentence $\mathbf{f}_t$ we have a new "best" translation $\mathbf{e}_t^{(r^*|\mathcal{E}_d)}$ where:

$$r_t^*|\mathcal{E}_d = \arg\max_r Q(\mathbf{e}_t^{(r)}|\mathcal{E}_d) \tag{3.11}$$

With the reference translations, we can calculate the gain in BLEU score when we use $\mathbf{e}_t^{(r^*|\mathcal{E}_d)}$ to replace $\mathbf{e}_t^{(1)}$, or in other words, the benefit of using corpus chunk $\mathcal{E}_d$ to rerank sentence $\mathbf{f}_t$ as:

$$\mathcal{G}(t,d) = BLEU(\ldots, \mathbf{e}_t^{(r^*|\mathcal{E}_d)}, \ldots) - BLEU(\ldots, \mathbf{e}_t^{(1)}, \ldots) \tag{3.12}$$

The higher the value $\mathcal{G}(t,d)$, the more suited $\mathcal{E}_d$ is to re-ranking the N-best translation list of $\mathbf{f}_t$. In other words, corpus chunk $\mathcal{E}_d$ is more *relevant* to sentence $\mathbf{f}_t$.

As an oracle study, we could calculate the gains over the whole testing set if we use the oracle most relevant corpus chunk to re-rank the N-best list. Table 3.5 shows the BLEU/NIST scores of the re-ranked best translation. The re-ranked system improved over the baseline model-best translation by about 10 BLEU points and was close to the oracle-best translation from the N-best list.

To answer the question whether "relevant" data is better than using all the data, the corpus chunks are incrementally added to build larger language models for re-ranking. Corpus chunks are added in the way that their relevance to a testing sentence are decreasing according to the oracle. In other words, we increase the data size but the data is becoming less and less relevant to the

	No postprocessing		With postprocessing	
	BLEU	NIST	BLEU	NIST
Model-best	31.92	8.22	33.25	8.33
95% Interval of random rerank				
Oracle-best	45.04	9.31	47.16	9.49
Re-ranked with the oracle corpus chunk	41.67	8.98	43.27	9.13

Table 3.5: Re-rank the N-best list using the most relevant corpus chunk known from the oracle

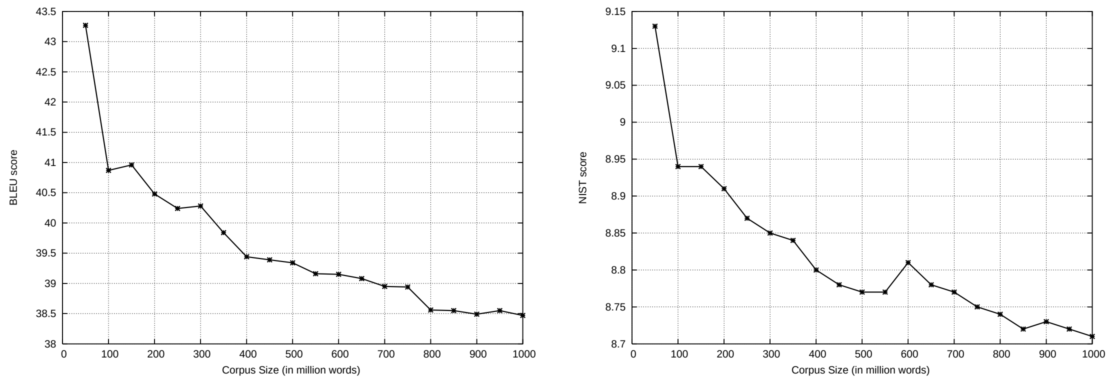


Figure 3-4: Oracle study: relevant data is better than more data.

testing data.

Disregarding some fluctuations in Table 3.6, it is clear that by adding more corpus chunks that are less relevant, both the BLEU and NIST scores of re-ranked translations decrease (Figure 3-4). This supports our argument that "relevant data is better than more data."

3.2.4 Selecting Relevant Data

We studied several simple statistics to select relevant corpus chunks to build LM for each testing sentence. These relevance statistics are based on the n-gram coverage of the corpus chunk on the N-best list.

Number of corpus chunks	Size (M words)	BLEU	NIST
1	50	41.67	8.98
2	100	39.34	8.78
3	150	39.35	8.79
4	200	38.87	8.75
5	250	38.66	8.71
6	300	38.72	8.71
7	350	38.33	8.70
8	400	37.78	8.65
9	450	37.71	8.63
10	500	37.66	8.62
11	550	37.51	8.62
12	600	37.47	8.65
13	650	37.26	8.63
14	700	37.12	8.61
15	750	37.17	8.60
16	800	36.70	8.58
17	850	36.93	8.58
18	900	36.87	8.59
19	950	36.92	8.58
20	1000	36.84	8.56

Table 3.6: Oracle study: relevant data is better than more data. Each corpus chunk has about 50M words.

n-gram Coverage Rate

Define the n-gram coverage rate of corpora $\mathcal{E}_d$ to sentence $\mathbf{f}_t$ as:

$$R(\mathcal{E}_d; \mathbf{f}_t) = \sum_{r=1}^{N} Q_c(\mathbf{e}_t^{(r)} | \mathcal{E}_d) \tag{3.13}$$

For each one of the N hypotheses, $Q_c(\mathbf{e}_t^{(r)} | \mathcal{E}_d)$ is the number of n-gram tokens in the hypothesis which can be found in $\mathcal{E}_d$ (see Eqn. 3.2). Informally, $R(\mathcal{E}_d; \mathbf{f}_t)$ estimates how well $\mathcal{E}_d$ covers n-grams (tokens) in the N-best list of $\mathbf{f}_t$. The higher the coverage, the more relevant $\mathcal{E}_d$ is to $\mathbf{f}_t$).

In the distributed LM architecture, the client first sends N translations of $\mathbf{f}_t$ to all the servers. From the returned n-gram matching information, the client calculates $R(\mathcal{E}_d; \mathbf{f}_t)$ for each server, and chooses the most relevant (e.g., 20) servers for $\mathbf{f}_t$. The n-gram counts returned from these relevant servers are summed up for calculating the likelihood of $\mathbf{f}_t$. As suggested by the oracle study in Table 3.6, one could also assign weights to the n-gram counts returned from different servers during the summation so that the relevant data has more impact than the less-relevant ones.

Figure 3-5 shows the results of N-best list reranking using different metrics. The corpus chunks used to build the LM are added incrementally according to their relevance to the testing data calculated by equation 3.13. The selected data chunks may differ for different sentences. For example, the 2 most relevant corpora for sentence 1 are *Xinhua2002* and *Xinhua2003* while for sentence 2 *APW2003A* and *NYT2002D* are more relevant. When we use the most relevant data chunk (about 20 million words) to re-rank the N-best list, 36 chunks of data will be used at least once for 919 different sentences, which accounts for about 720 million words in total. Thus the x-axis in Figure 3-5 should not be interpreted as the total amount of the whole test set but the amount of corpora used for each sentence.

All three metrics in Figure 3-5 show that using all data together (150 chunks, 2.97 billion words) does not give better discriminative powers than using only some relevant chunks. This supports our argument that relevance selection is helpful in N-best list re-ranking. In some cases the re-ranked N-best list has a higher BLEU score after adding a supposedly "less-relevant" corpus chunk and a lower BLEU score after adding a "more-relevant" chunk. This indicates that the relevance measurement (Eq. 3.13) does not fully reflect the real "relevance" of a data chunk for a sentence. With a better relevance measurement one would expect to see the curves in Figure 3-5 to be much smoother.

3.2.5 Weighted n-gram Coverage Rate

Unlike the unweighted n-gram coverage rate where n-grams of different orders are considered equally, weighted n-gram coverage rate applies different weights on n-gram coverage rates of different orders. Intuitively, we should predict one a corpus chunk to be more relevant than another one if has a higher high-order n-gram coverage rate. Define $M(\mathcal{E}, \mathbf{f})$, the *weighted n-gram coverage rate* of corpus $\mathcal{E}$ for a sentence $\mathbf{f}$ as:

$$M(\mathcal{E}, \mathbf{f}) = \sum_{\mathbf{k=1}}^{\mathbf{n}} \lambda_{\mathbf{k}} \text{Coverage}_{\mathbf{k}}(\mathcal{E}, \mathbf{f}) \tag{3.14}$$

where $\text{Coverage}_k(\mathcal{E}, \mathbf{f})$ is the k-th order n-grams coverage rate of the corpus on $\mathbf{f}$'s N-best list:

$$\text{Coverage}_k(\mathcal{E}, \mathbf{f}) = \frac{\sum_{\mathbf{e_i^{i+k-1}} \in \{\mathbf{e_t^{(1)}} \ldots \mathbf{e_t^{(N)}}\}} \mathbf{C}(\mathbf{e_i^{i+k-1}}) \mathbf{\Phi}(\mathbf{e_i^{i+k-1}}, \mathcal{E})}{\sum_{\mathbf{e_i^{i+k-1}} \in \{\mathbf{e_t^{(1)}} \ldots \mathbf{e_t^{(N)}}\}} \mathbf{C}(\mathbf{e_i^{i+k-1}})}, \tag{3.15}$$

$C(e_i^{i+k-1})$ is the frequency of n-gram e_i^{i+k-1}'s occurrence in the N-best list and

$$\Phi(e_i^{i+k-1}, \mathcal{E}) = \begin{cases} 1, & \text{if } e_i^{i+k-1} \text{exists in } \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \tag{3.16}$$

Informally, we predict a corpus chunk $\mathcal{E}_d$ to be mostly "relevant" to a testing sentence $\mathbf{f}$ if n-grams in $\mathbf{f}$'s N-best translations can be mostly covered by $\mathcal{E}_d$.

The weights λ_k are optimized over the dev-test set. The experiment is set up as follows: The LDC Gigaword corpus (both version 2 and version 3) of 4,916,610,128 words are split into 104 chunks based on their natural order (news source and time released). The N-best list used in this experiment is a list of 1000-best translations of 191 testing sentences extracted from the Chinese MT06 test set by the Hiero system. For each one of the 191 testing sentence, we use each individual corpus chunk to rerank its N-best list using the Q_{nc} feature. The difference of the reranked-best hypothesis and the model-best hypothesis in BLEU score is the *gain* of the corpus chunk denoted as $\mathcal{G}(t, d)$ (definition 3.12). $\mathcal{G}(t, d)$ can be negative if the reranked-best hypothesis has lower BLEU score than the model-best baseline translation. We also estimate the relevance feature of each corpus for each of the testing sentence, such as the 1-gram coverage rate, 2-gram coverage rate, etc. Relevance predicting then becomes a typical linear prediction problem where we want to optimize the linear weights λ_k such that the linear combination of various relevance features (k-gram coverage rate) could best predict the *gain* of a corpus chunk for a testing sentence. Given the training data of $191 \times 104 = 19864$ data points, we use the Nelder-Mead method (also known as the downhill simplex method) [Nelder and Mead, 1964] to find the optimal λ_k.

Nelder-Mead method stops at a local maximum for each starting point. To understand the usefulness of each relevance feature, we ran the downhill simplex optimization several thousand times, each with a random starting point. We estimate the percentage of local maximum that have positive gains from all random starts. Since we could only use one set of weights to predict the relevance of a corpus chunk for unseen sentences, this estimation has no practical meaning. However, given the nature of the rugged search space, it is interesting to see which relevance feature is useful in identifying relevant corpus chunks if we could explore the search space "thoroughly".

Table 3.7 shows the effectiveness of using different relevance features and their combinations to choose the most relevant corpus for reranking. For each testing sentence, there are 104 pairs of <features (vectors), predicted gain>. With the tuned feature weights, we select the corpus chunk with the highest combined feature value and use the associated gain value as the predicted reranking gain. The sum of the predicted gain over all 191 sentences is the predicted gain for reranking the whole test set. For each of the 5000 random starts, we use downhill simplex optimization to find a locally maximum gain. Report in table 3.7 are the maximum gains of all random starts (Max Bleu), the percentage of random starts that end up in a local maximum with a positive gain, the average gain value of the positive gains, the percentage of random starts that end up in a local maximum with a negative gain (reranked translations have lower BLEU scores than the baseline) and the average gain value of those cases end up in negative.

Features used in the experiment include single features such as the unweighted n-gram coverage rate (percentage of n-grams of all orders in the N-best list that can be found in the corpus chunk), unigram, bigram, trigram, 4-gram, 5-gram and 6-gram coverage rate, the averaged 6-gram conditional probabilities of the N-best list $Q_{l(6)}$, sum of the non-compositionalities and the averaged non-compositionalities (Q_{nc}). Features are also combined to estimate $M(\mathcal{E}, f)$. Shown in the table are results of combining n-gram coverage rate of 1,2,3-gram; 1,2,3,4-gram;1,2,3,4,5-gram; 2,3,4,5-gram and 3, 4, 5, 6-grams. As table 3.7 shows, in general higher order n-gram coverage is more powerful in selecting relevant corpus chunks than lower-order n-grams. The unweighted n-gram coverage $R(\mathcal{E}; f)$ disregard the order of the n-gram is not as discriminative as the weighted n-gram coverage rate. Combining n-gram coverage of different order, especially with higher orders best

Relevance Feature (Combination)	Max BLEU Gain	Random starts end up with positive gains	Average positive BLEU gain	Random starts end up with negative gains	Average negative BLEU gain
$R(\mathcal{E};\mathbf{f}) = \sum_{r=1}^{N} Q_c(\mathbf{e}^{(r)}\|\mathcal{E})$	0.97	71.82%	0.97	28.16%	-1.22
$\text{Coverage}_1(\mathcal{E};\mathbf{f})$	0.20	72.85%	0.01	27.12%	-1.51
$\text{Coverage}_2(\mathcal{E};\mathbf{f})$	1.40	74.99%	0.72	24.99%	-1.36
$\text{Coverage}_3(\mathcal{E};\mathbf{f})$	1.52	76.84%	0.48	23.14%	-1.20
$\text{Coverage}_4(\mathcal{E};\mathbf{f})$	1.33	74.55%	1.33	25.43%	-1.87
$\text{Coverage}_5(\mathcal{E};\mathbf{f})$	2.07	74.40%	2.06	25.58%	-1.71
$\text{Coverage}_6(\mathcal{E};\mathbf{f})$	1.97	75.71%	1.86	24.27%	-0.76
$\frac{1}{N}\sum_{r=1}^{N} Q_{l(6)}(\mathbf{e}^{(r)}\|\mathcal{E})$	1.06	73.37%	1.06	27.61%	-1.21
$\sum_{r=1}^{N} Q_{nc}(\mathbf{e}^{(r)}\|\mathcal{E})$	0.00	0%	0.00	99.98%	-0.37
$\frac{1}{N}\sum_{r=1}^{N} Q_{nc}(\mathbf{e}^{(r)}\|\mathcal{E})$	0.00	0%	0.00	99.98%	-0.24
$\text{Coverage}_k(\mathcal{E};\mathbf{f})$					
$k = 1, 2, 3$	2.13	59.44%	1.23	40.52%	-1.15
$k = 1, 2, 3, 4$	2.44	55.92%	1.51	44.04%	-1.20
$k = 1, 2, 3, 4, 5$	**2.79**	54.49%	1.73	45.47%	-1.14
$k = 1, 2, 3, 4, 5, 6$	**2.79**	59.72%	1.61	40.24%	-1.03
$k = 2, 3, 4, 5$	2.62	53.25%	1.69	46.71%	-1.10
$k = 3, 4, 5, 6$	2.71	67.20%	1.71	32.75%	-0.98

Table 3.7: Effectiveness of different relevance features (and their combinations).

predicts the corpus' relevance.

Table 3.8 shows the optimized weights for each relevance feature combinations. The feature space seems to be quite bumpy and the optimized feature weights lack generalization. For example, if the feature weights are reasonably generalized, one may draw the conclusion that the 1-gram coverage rate correlates with the corpus relevance in a negative way since its optimized weights in the first three feature combinations are all negative. However in the case with 1, 2, 3, 4, 5, and 6-gram coverage rate, the optimized weight for 1-gram is positive (53.91).

Table 3.9 shows the gain in BLEU score after reranking the N-best list using the most relevant corpus chunk selected based on manually set feature weights. Feature weights are set according to the heuristic that higher order k-gram coverage rates should be given more weights since they have more discriminative power in selecting relevant data than lower k-grams. Even though the BLEU gains in table 3.9 are lower than those based on optimized feature weights, they are more meaningful and interpretable.

	Optimized Weights of k-gram Coverage Rate						
Feature Combinations	1	2	3	4	5	6	BLEU Gain
1, 2, 3-gram coverage rate	-93.61	59.06	-2.06				2.13
1, 2, 3, 4-gram coverage rate	-39.61	17.01	-3.55	4.95			2.44
1, 2, 3, 4, 5-gram coverage rate	-51.77	5.98	-14.99	-6.68	86.13		2.79
1, 2, 3, 4, 5, 6-gram coverage rate	53.91	96.83	-54.46	10.97	137.23	-54.15	2.79
2, 3, 4, 5, 6-gram coverage rate		4.64	-8.38	-7.43	66.29		2.62
3, 4, 5, 6-gram coverage rate			7.61	-9.71	-4.42	40.00	2.71

Table 3.8: Optimized weights in k-gram coverage rate combination.

	Weights of k-gram Coverage Rate						
Feature Combinations	1	2	3	4	5	6	BLEU Gain
1, 2-gram coverage rate	1.0	2.0					0.73
1, 2-gram coverage rate	-1.0	2.0					0.77
1, 2-gram coverage rate	-2.0	1.0					1.12
1, 2, 3-gram coverage rate	-2.0	1.0	2.0				0.74
1, 2, 3, 4-gram coverage rate	-2.0	1.0	2.0	3.0			1.27
1, 2, 3, 4, 5-gram coverage rate	-2.0	1.0	2.0	3.0	4.0		1.30
2, 3, 4, 5-gram coverage rate		2.0	3.0	4.0	5.0		1.11
2, 3, 4, 5-gram coverage rate		4.0	9.0	16.0	25.0		**1.54**
2, 3, 4, 5, 6-gram coverage rate		2.0	3.0	4.0	5.0	6.0	1.07
2, 3, 4, 5, 6-gram coverage rate		4.0	9.0	16.0	25.0	36.0	1.14
1, 2, 3, 4, 5, 6-gram coverage rate	-2.0	2.0	3.0	4.0	5.0	6.0	1.09
1, 2, 3, 4, 5, 6-gram coverage rate	1.0	4.0	9.0	16.0	25.0	36.0	1.14

Table 3.9: Feature weights set following the heuristic that higher order k-gram coverage rates have more discriminative power in choosing relevant corpus chunks.

3.2.6 Related Work

Selecting corpus to train the language model for a particular testing sentence or document is known as the *language model adaptation*. [Bellegarda, 2004] provides a very good review of the major language model adaptation approaches in speech recognition research. A typical approach, such as the one described in [Iyer and Ostendorf, 1999], selects training data based on its relevance for a topic or the similarity to data known to be in the same domain as the test data. Each additional document is classified to be in-domain or out-of-domain according to cosine distance with $TF-IDF$ term weights, POS-tag LM and a 3-gram word LM. n-gram counts from the in-domain and the additionally selected out-of-domain data are then combined with an weighting factor. The combined counts are used to estimate a LM with standard smoothing. LM adaptation has also been applied in SMT. For example, the work presented in [Zhao et al., 2004] uses structured query composed from the N-best list to retrieve documents that are relevant to the testing data and builds an adapted LM for another round of translation.

Most of the LM adaptation work requires to train an adapted LM offline which can not be done in real time. In the distributed LM system, we can dynamically select the relevant corpus chunks for each sentence on the fly.

3.2.7 Experiments

Table 3.10 lists results of the re-ranking experiments under different conditions. The re-ranked translation improved the BLEU score from 31.44 to 32.64, significantly better than the model-best translation.

Different metrics are used under the same data situation for comparison. Q_c, though extremely simple, gives quite nice results on N-best list re-ranking. With only one corpus chunk (the most relevant one) for each source sentence, Q_c improved the BLEU score to 32.27. We suspect that Q_c works well because it is inline with the nature of BLEU score. BLEU measures the similarity between the translation hypothesis and human reference by counting how many n-grams in MT can be found in the references.

Instead of assigning weights 1 to all the matched n-grams in Q_c, Q_{nc} weights each n-gram by its *non-compositionality*. For all data conditions, Q_{nc} consistently gives the best results.

Metric family $Q_{l(n)}$ is close to the standard n-gram LM probability estimation. Because no smoothing is used, $Q_{l(3)}$ performance (32.00) is slightly worse than the standard 3-gram LM result (32.22). On the other hand, increasing the length of the history in $Q_{l(n)}$ generally improves the performance.

# of Relevant Chunks per. Sent	1	2	5	10	20	150
offline 3-gram KN	32.22					32.08
offline 4-gram KN	32.22					32.53
Q_c	32.27	32.38	32.40	32.47	32.51	32.48
$Q_{l(3)}$	32.00	32.14	32.14	32.15	32.16	
$Q_{l(4)}$	32.18	32.36	32.28	32.44	32.41	
$Q_{l(5)}$	32.21	32.33	32.35	32.41	32.37	
$Q_{l(6)}$	32.19	32.22	32.37	32.45	32.40	32.41
$Q_{l(7)}$	32.22	32.29	32.37	32.44	32.40	
Q_{nc}	32.29	32.52	**32.61**	32.55	**32.64**	32.56

Table 3.10: BLEU scores of the re-ranked translations. Baseline score = 31.44

System	LM Size	LM Type	Bleu	NIST
Baseline	200 M	3-gram	**31.44**	9.01
Re-ranked	2900 M	3-gram	32.08	9.14
		4-gram	**32.53**	9.19
	20M for each Sent	4-gram	32.22	9.14
		Q_{nc} PMI Adjacent n-grams	32.34	9.20
		Q_t: PMI Phrasal Triggers	32.46	9.21
		$Q_{nc} + Q_t$	32.60	9.24
	200M for each Sent.	Q_{nc}	**32.60**	9.23
		Q_t	**32.79**	9.25
		$Q_{l(6)} + Q_t$	**32.90**	9.28

Table 3.11: Reranking N-best list using distributed language model.

3.3 DB Language Model

During the GALE project, Google and Yahoo! released n-gram statistics collected from very large web corpus to the research community for language model training. The Google LM was based on 5-grams extracted from 1 terawords of web documents covering a wide range of sources. To limit the size of the distribution, only N-grams occurring 40 times or more were included; still it contained 0.31G bigrams, 0.98G trigrams, 1.3G 4-grams and 1.2G 5-grams, over a vocabulary of 11M words, and occupied about 25G on disk even after file compression. The Yahoo n-gram data is extracted from about 3.4G words of news sources during a 1-year period prior to the epoch of the GALE 2007MT evaluation data. Although containing all n-grams occurring more than once, this collection

n	Num. of Types	DB Size
1-gram	13,588,391	446 MB
2-gram	314,843,401	12.0 GB
3-gram	977,069,902	41.94 GB
4-gram	1,313,818,354	64.41 GB
5-gram	1,176,470,663	65.20 GB

Table 3.12: Google n-gram data and the size of indexed database on disk.

was much smaller, but also more recent and directly focused on the target domain broadcast news) than the Google corpus. It comprised about 54M bigrams, 187M trigrams, 288M 4-grams, and 296M 5-grams. The Yahoo n-gram contains information which is needed to calculate probabilities using the modified KN smoothing [Kneser and Ney, 1995]. However, due to the problem with corrupted data in the released DVD, we could not train the LM using the Yahoo n-grams.

Both Google and Yahoo data come with only n-gram statistics rather the original corpus. Thus we can not use the Suffix Array Language Model to index the corpus and calculate the LM probability in the distributed framework. To build an LM that avoids memory issues we implemented a Data-base Language Model (DBLM). Berkeley-DB (BDB) [Olson et al., 1999] is an Open Source embedded database system. It provides functionalities to store and access $\langle key, value \rangle$ pairs efficiently. Large databases can only be stored on disk, smart caching is implemented internally in BDB to speed up the database access. The n-gram statistics are stored as $\langle n - gram, frequency \rangle$ pair in BDB.

With n-gram frequencies provided by the DB, we use the Jelinek-Mercer deleted interpolation smoothing approach [Jelinek and Mercer, 1980] to calculate the language probability of a word given its $n - 1$ word history. Interpolation weights were estimated on the held-out tuning set.

We used the DBLM to calculate the probability for each of the hypothesis in the N-best list.

The speed of DBLM on N-best is acceptable. However, it is too slow to be used in the statistical machine translation decoder to make use of such large n-gram data. The decoder needs to query the LM for millions of times when translating a testing sentence. Even with the DB's internal cache and an additional cache in the decoder, DBLM still needs huge number of disk I/O to access the n-gram frequencies to calculate the probability.

3.4 Related Work

Similar distributed language models are developed independently by [Emami et al., 2007] and [Brants et al., 2007] around the same time.

To make use of large training data, [Soricut et al., 2002] build a Finite State Acceptor (FSA) to compactly represent all possible English translations of a source sentence according to the translation model. All sentences in a big monolingual English corpus are then scanned by this FSA and those accepted by the FSA are considered as possible translations for the source sentence. The corpus is split into hundreds of chunks for parallel processing. All the sentences in one chunk are scanned by the FSA on one processor. Matched sentences from all chunks are then used together as possible translations. The assumption of this work that possible translations of a source sentence can be found as exact match in a big monolingual corpus is weak even for very large corpus. This method can easily fail to find any possible translation and return zero proposed translations.

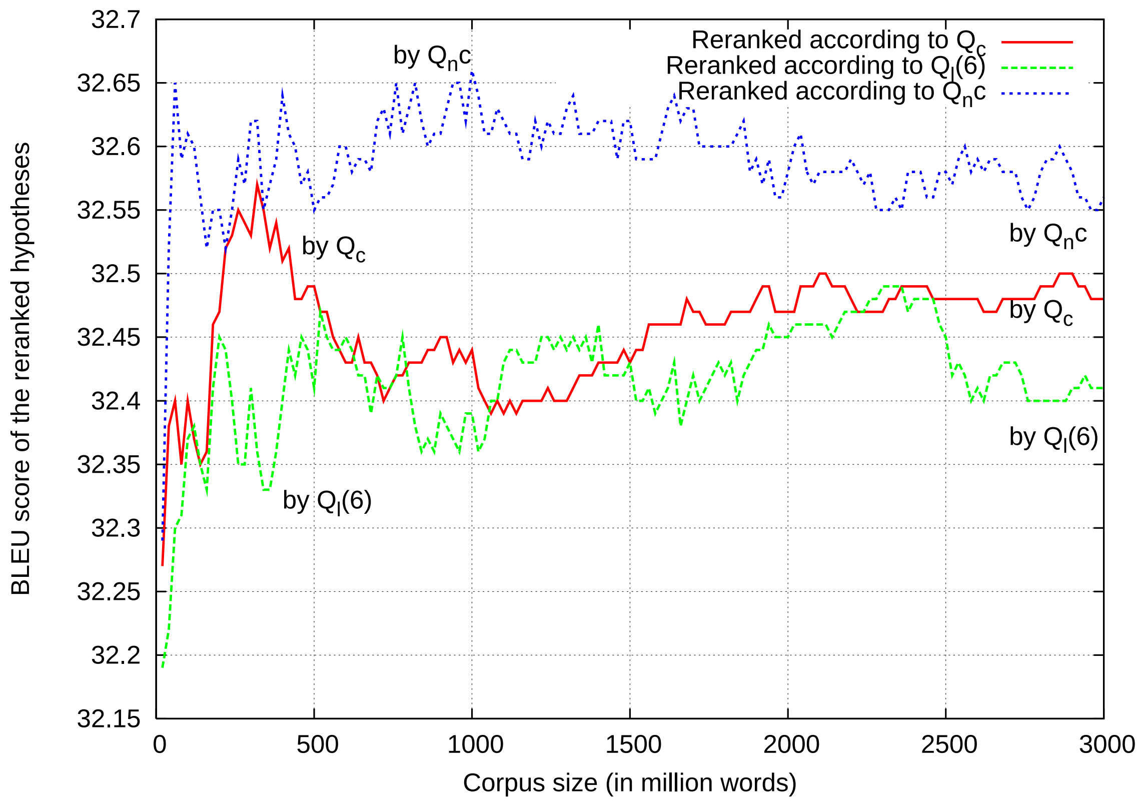


Figure 3-5: BLEU score of the re-ranked best hypothesis vs. size of the relevant corpus (in million words) for each sentences. Three reranking features are shown here: Q_c (number of n-grams matched), $Q_{l(6)}$ (interpolated 6-gram conditional probability averaged on length) and Q_{nc} (sum of n-grams' non-compositionality)

.

Chapter 4

Discriminative Language Modeling

n-gram language models are generative models. An n-gram language model tries to model the generative process of a sentence given the training data of correct sentences. The parameters in the n-gram models are optimized to maximize the likelihood of generating the training data. One assumption in the generative language models is that any word sequence is possible, i.e. with a probability greater than 0, or in other words, generative language models do not explicitly modeling the incorrect outcome. However, there are certain phenomena in natural languages which we know should never happen. For example, "the" should never occur at the end of an English sentence; in Japanese, particles "wa" and "ga" should never be placed at the beginning of a sentence [1].

In the case of the generative n-gram LM, it has never seen such events in the training data since it is trained only from those "correct" examples. n-gram LM uses various smoothing techniques to estimate the probability for such unseen events and hopefully it will assign low probabilities to them. But assigning a low probability to "the </s>" or "<s> ga" can not prevent them from being generated by the SMT decoder.

In this chapter, we describe the discriminative language modeling approach to explicitly model

[1] According to the Japanese grammar (http://www.timwerx.net/language/particles.htm), *wa* and *ga* indicate subjects by coming after them. *wa* is the "standard" subject indicator. It indicates the general topic and, if anything, emphasizes what comes after it, such as *Nihon no natsu wa atsui desu. (Summers in Japan are hot.) Ga* is used with simple question subjects in many cases, such as *Dare ga kono gyuunyuu o koboshita? (Who spilled this milk?* and *Nani ga tabetai no? (What do you want to eat?).*

and penalize incorrect n-grams that could be generated by a SMT decoder.

4.1 Discriminative Language Model

The process of discriminative language model training starts with using the current SMT models to translate the source side of the training data. Unlike the EBMT system which would generate the exact target side of the bilingual training data, the phrase-based SMT systems usually output different sentences (we refer them as hypotheses) compared with the target side of the training data (references). The key idea of the discriminative language model is thus to train a model that could correct the difference between the hypotheses and the references. By applying the discriminative language model, we try to push the n-gram distribution in the hypotheses closer to the distribution in the references.

4.2 Perceptron Learning

The concept of the original perceptron method [Rosenblatt, 1958] is a linear classifier:

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{else.} \end{cases} \tag{4.1}$$

Learning is modeled as the update of the weight vector after each iteration on data points where the output y is different from the desired output δ:

$$w(i)' = w(i) + \alpha(\delta - y)x(i) \tag{4.2}$$

[Collins, 2002] uses the perceptron algorithm to train a discriminative part-of-speech (POS) tagger. The training algorithm starts by tagging the labeled training corpus using a model with randomly initialized parameters. The best tagged sequences for each sentence using the current model (*hyp*) are then compared with their corresponding true labels in the training data (*ref*). For every tag trigram type seen c_1 times in *hyp* and c_2 times in *ref*, update the weight of this trigram to $w \leftarrow w + c_1 - c_2$. For every tag/word type seen c_3 times in *hyp* and c_4 times in *ref*, update the weight of this pair to $w \leftarrow w + c_3 - c_4$.

Following the same procedure, we use the perceptron learning algorithm to train a discriminative language model for the translation system. Given the current translation model and a generative language model, we translate the source side of the bilingual training corpus f into e'. We enumerate all n-grams from the union of e and e'. For each n-gram types, we increase the n-gram's weight if its frequency in e' is less than its frequency in e and decrease its weight if it has been over generated. The adjusted weights for n-grams are then used as additional feature functions in the decoder. Combining with other decoder features in a log-linear model, we apply the discriminative language model to translate the training corpus and repeat the above procedure to update the discriminative langauge model.

In other words, we iteratively adjust the weight of each n-gram in the discriminative language model to push the generated translation results towards the reference translation.

4.3 Implementation and Experiments

The most time-consuming part of the discriminative language model training process is to translate the whole training corpus with the current/updated models. The discriminative language model is an error-correction model for a running translation system and the learned DLM is specific to errors made by the baseline translation system. In other words, a discriminative language model trained for a small-scale baseline system may not help another baseline system which uses larger models. By using multiple large generative LMs and large phrase translation lists in the decoder, we usually get better translation results. As we want to apply the discriminative language model to improve the best MT system, we need to use the best MT system as the baseline to train the discriminative language model. This means that we need to use the full-scale MT system to translate the whole training corpus. The translation process requires enormous amount of time for news wire type of training data [2]. To test the discriminative language model in the real system, we trained and tested the DLM on the Basic Travel Expression Conversation (BTEC) data [Takezawa et al., 2002].

[2]Assuming that the decoder translates one sentence per second using the full-scale system, translating a 200M word training corpus requires 2700 hours for one iteration.

n-gram	Freq. in Hyp.	Freq in Ref.
it 's 二年	11	0
<s> 四人	23	0
when is the same	24	0
this ten one dollar	11	0
twenty minute < /s>	27	0
when will be	10	0
can I have not	13	0
I am I	31	0

Table 4.1: Example of n-gram types in MT output but never occur in the reference (target side of the training data).

BTEC contains 162K sentence pairs and on average each sentence has about 7 words. Translating the whole corpus on one CPU takes about 10 hours for one iteration. Split the corpus and translate each chunk on a computer cluster with N nodes cuts down the time to $1/N$.

Table 4.1 lists some n-gram types in the MT output which do not exist in the reference at all. n-grams containing untranslated source words such as " it 's 二年 " obviously do not exist in the reference. Penalizing these n-grams by the discriminative language model may have limited or no impact since the translation model does not have other translations for these words. In this particular experiment, those words are not translated due to mistakes in the number tagger. In other cases, penalizing n-gram types like *this ten one dollar* and *can I have not* through the discriminative language model can effectively reduce their occurrences in the translation. Similar cases are n-gram types which are over generated (Table 4.3), not generated by the MT system at all (Table 4.2) and under-generated (Table 4.4).

Table 4.5 shows the discrepancy between the reference translation and the MT output during the process of discriminative language model training. The discrepancy is measured by the sum of frequency difference for all n-gram types (up to order 3). BLEU scores of 1000 sentences from the training data improve over iterations. BLEU scores of the dev-set initially improve with the updated DLM, but decrease after the fourth iteration due to the overfitting on the training set.

Table 4.6 shows the performance of using the discriminative language model for both Japanese to English and English to Japanese translation directions on the dev-set and unseen testing data.

n-gram	Freq. in Hyp.	Freq in Ref.
like three	0	16
get to @PLACE.landmark < /s>	0	25
name 's @PERSON.firstname	0	16
in @DATE.month < /s>	0	107
the phone < /s>	0	10
back at @TIME < /s>	0	17
cents < /s>	0	77
right now < /s>	0	13

Table 4.2: Example of n-gram types in the reference translation but are not generated in the MT output.

n-gram	Freq. in Hyp.	Freq in Ref.
you concentrate	207	1
few few	109	1
<s> do not know	96	1
a few few	95	1
and .	165	3
the . < /s>	108	2
and . < /s>	156	3
my . < /s>	49	1
<s> and also	86	2
a few < /s>	43	1

Table 4.3: Example of n-gram types that are over-generated by the MT system.

n-gram	Freq. in Hyp.	Freq in Ref.
@PLACE.country < /s>	1	1667
@NUMBER.sequence < /s>	1	659
@NUMBER < /s>	2	636
@PERSON.lastname < /s>	2	484
@DATE < /s>	2	426
years ago < /s>	1	159
dollars < /s>	5	526
United States . < /s>	1	64
Japan ? < /s>	1	62
Canada	2	115

Table 4.4: Example of n-gram types that are under-generated by the MT system.

Update Iteration	$\sum_{\tilde{e}} \lvert C_{\text{hyp}}(\tilde{e}) - C_{\text{ref}}(\tilde{e}) \rvert$	BLEU on Training	BLEU on Dev-test
		57.61	55.87
1	849,138	58.90	55.46
2	826,716	59.06	56.03
3	799,574	59.14	56.35
4	771,872	59.46	56.11
9	702,564	59.98	56.01

Table 4.5: 9 iterations of discriminative language model training. The discrepancy between the reference translation and the MT hypothesis on the training data becomes smaller with the updated discriminative language model.

	Training		Testing	
	w/o DLM	with DLM	w/o DLM	with DLM
J $\rightarrow$ E	58.90	60.01	58.64	58.13
E $\rightarrow$ J	59.40	60.51	46.40	47.01

Table 4.6: Impact of a discriminative language model on the Japanese/English translation system.

On the training data set DLM pushes the generated translation towards the reference translation (the target side of the bilingual corpus) and the BLEU scores are improved. However, DLM slightly over-fits the training and does not show the same improvement over the testing data. On the other hand, when we subjectively evaluate the translations generated with/without the DLM, human subjects prefer the translation generated using the DLM. One explanation to this is that BLEU score is not so sensitive to phenomena such as Japanese particles occur at the beginning of the sentence, but correcting such errors make the sentence much more readable to humans.

4.4 Limitation of the Discriminative Language Model

The training of the discriminative language model relies on the availability of bilingual data. The decoder uses the current model to translate the source side of the bilingual data and compares the translation with the target side to train the discriminative language model. Compared to the size of the available monolingual data, the bilingual data is much smaller. Available bilingual data is in the range of several hundred million words even for very high density language pairs like Chinese/English and Arabic/English. The coverage of the discriminative language model is thus much smaller compared to the generative language model trained from tens or hundreds of billions of words. The issue of over-fitting also arises due to limited training data.

4.5 Related Work

[Stolcke and Weintraub, 1998] experimented with various discriminative LM training approaches based on maximum mutual information estimation and gradient descent in the LM parameter space. These approaches were largely unsuccessful because of data sparseness and overfitting problems. In [Stolcke et al., 2000], a more robust approach of using a separate "anti-language model" was tried. The idea is to construct a separate "anti-LM" to penalize likely mis-recognitions. n-grams from the recognition hypothesis were weighted by the posterior probabilities of the hypotheses in which they occurred. n-grams with a total expected count of at least 1 were used in estimating a backoff trigram anti-LM. In other words, if an n-gram always occur in hypotheses with low posterior probabilities, it is likely that this n-gram is "bad" and contributes to the incorrectness of the recognition and thus needs to have a lower probability. The "anti-LM" is later combined with the standard acoustic and language models in a log-linear model. [Stolcke et al., 2000] reported marginal gain on word error rate after applying the anti-language model. [Chen et al., 2000] presented a method based on changing the trigram counts discriminatively, together with changing the lexicon to add new words. [Roark et al., 2004] compared two parameter estimation methods for discriminative language model training: the perceptron algorithm, and a method based on conditional random fields (CRFs). The perceptron algorithm automatically select a relatively small feature set after a couple of passes over the training data. Using the feature set CRF output from the perceptron algorithm, CRF training provides an addition 0.5% reduction in word error rate and in total achieved 1.8% absolute reduction in WER by applying the discriminative language model. End-to-end discriminative training has been shown to improve the translation quality in [Liang et al., 2006].

Chapter 5

x-gram Language Models

The binding of words and phrases to form hierarchically organized constituent structures is a property shared by a wide variety of linguistic models that differ in many other respects and it plays a crucial role in explanations of many linguistic phenomena. Long-distance dependencies found in wh-questions, topicalization, relativization, passivization, raising, and scrambling structures consistently involve the appearance of a constituent in a non-canonical position. It is difficult to capture such rules without constituents.

Interpreted from the constituent parsing perspective, the dependency parsing is a repeating process of segmenting the sentence into constituents and selecting one word as the head word for each constituent to form a more abstract sentence.

Given the dependency parse of sentence *Russia's proposal to Iran to establish a joint uranium enrichment enterprise within Russia 's borders was still effective* (Figure 5-1), word *was* can be better predicted by its constituent context *proposal* than its immediate n-gram context *'s borders.* In a 128M words corpus, the 3-gram log probability of *was* given *'s borders* is -2.15 where as $logP(was|proposal) = -0.699$. There are only two occurrences of the 3-gram *'s borders was* in the data (Figure 5-2).

It is clear that word *was* in these sentences are not triggered by its immediate context *'s borders* as would be the case in the n-gram model.

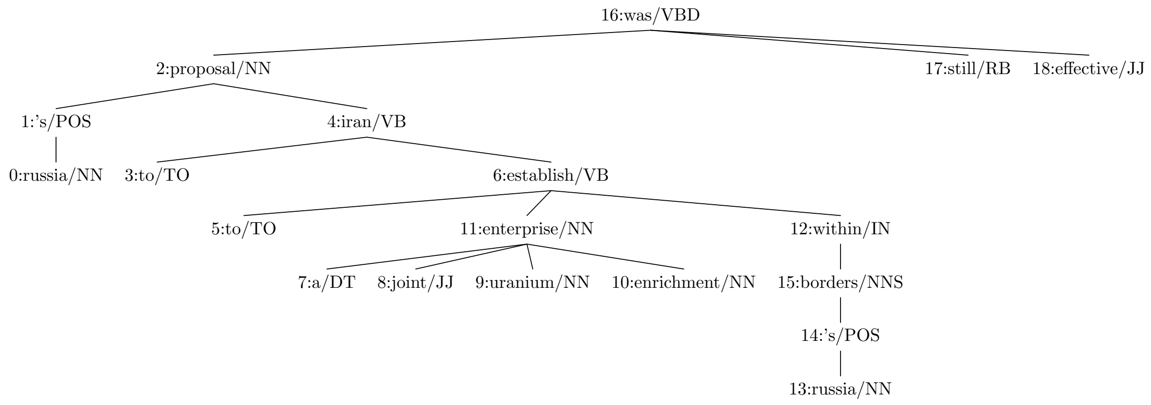


Figure 5-1: Dependency parse tree of sentence *russia 's proposal to iran to establish a joint uranium enrichment enterprise within russia 's borders was still effective*

SARS deputy director Tripmaker said last week that illegal goods across
South Africa **'s borders was** estimated at 30 billion dollars a year .

Second level of alertness concerning illegal infringement of
Bulgaria **'s borders was** imposed in the country , local media reported Monday .

Figure 5-2: Two occurrences of 3-gram *'s borders was* in a 128M words corpus.

5.1 Dependency Structure

5.1.1 Dependency Grammar

Dependency grammars [Hudson, 1984, Mel'čuk, 1988] are rooted on the concept of *dependency*. The fundamental notion of dependency is based on the idea that the syntactic structure of a sentence consists of binary asymmetrical relations between the words of the sentence [Nivre, 2005]. According to [Mel'čuk, 1988], the word forms of a sentence can be linked by three types of dependencies: *morphological*, *syntactic* and *semantic*.

Various dependency grammars have been proposed in recent years. They share common dependency assumptions for natural language and differ in specific features each grammar aims to capture.

5.1.2 Dependency Graph

The dependency structure of a sentence can be described as a graph [McDonald and Nivre, 2007]: Let $L = l_1, \ldots, l_{|L|}$ be a set of permissible arc labels. Let $x = w_0, w_1, \ldots, w_n$ be an input sentence where $w_0 = \mathit{root}$ and $w_1, \ldots, w_n$ are proper words. Formally, a dependency graph for an input sentence x is a labeled directed graph $G = (V, A)$ consisting of a set of nodes V and a set of labeled directed arcs $A \subseteq V \times V \times L$, i.e., if $(i, j, l) \in V$ and $l \in L$, then there is an arc from head word node i to child word node j with label l in the graph. A dependency graph G for sentence x must satisfy the following properties:

1. $V = 0, 1, \ldots, n$
2. If $(i, j, l) \in A$, then $j \neq 0$.
3. If $(i, j, l) \in A$, then for all $i' \in V - i$ and $l' \in L$, $(i', j, l') \notin A$.
4. For all $j \in V - 0$, there is a (possibly empty) sequence of nodes $i_i, \ldots, i_m \in V$ and labels $l_1, \ldots, l_m$, $l \in L$ such that $(0, i_1, l_1)$, (i_1, i_2, l_2),…, $(i_m, j, l_m) \in A$.

The constraints state that the dependency graph spans the entire input (1); that the node 0 is a root (2); that each node has at most one incoming arc in the graph (single-head constraint) (3); and that the graph is connected through directed paths from the node 0 to every other node in the graph (connectedness) (4). Also, by default, we pose the constraint on the graph that the graph should not contain cycles (acyclicity). A dependency graph satisfying these constraints is a directed tree originating out of the root node 0. We say that an arc (i, j, l) is *non-projective* if not all words k occurring between i and j in the linear order are dominated by i (where dominance is the transitive closure of the arc relation).

For a given dependency graph $G = (V, A)$, introducing a mapping function $\mathbf{h}$ such that $\mathbf{h}(j) = i$ iff. $(i, j, l) \in A$. In other words, $\mathbf{h}(j)$ returns the parent node of j. This parent node is the head word of the constituent where j is in. Denote the d-th dependency ancestor of node j as $\overbrace{\mathbf{h}(\mathbf{h}(\cdots \mathbf{h}}^{d}(j) \cdots)$, or $\mathbf{h}^d(j)$ in short.

5.2 Dependency Parsing

5.2.1 Global, Exhaustive, Graph-based Parsing

For an input sentence, $x = w_0, w_1, \ldots, w_n$, consider the dense graph $G_x = (V_x, A_x)$ where:

1. $V_x = 0, 1, \ldots, n$
2. $A_x = (i, j, l) | \forall i, j \in V_x$ and $l \in L$

Let $D(G_x)$ represent the subgraphs of graph G_x that are valid dependency graphs for the sentence x. Since G_x contains all possible labeled arcs, the set $D(G_x)$ must necessarily contain all valid dependency graphs for x.

Assume that there exists a dependency arc scoring function, $s : V \times V \times L \to \mathbb{R}$. Furthermore, define the score of a graph as the sum of its arc scores,

$$s(G = (V, A)) = \sum_{(i,j,l) \in A} s(i, j, l) \tag{5.1}$$

The score of a dependency arc, $s(i, j, l)$, represents the likelihood of creating a dependency from word w_i to w_j with the label l. If the arc score function is known a priori, then the parsing problem can be stated as,

$$G = \operatorname*{arg\,max}_{G \in D(G_x)} s(G) = \operatorname*{arg\,max}_{G \in D(G_x)} \sum_{(i,j,l) \in A} s(i, j, l) \tag{5.2}$$

The problem is equivalent to finding the highest scoring directed spanning tree in the graph G_x originating out of the root node 0, which can be solved for both the labeled and unlabeled case in $O(n^2)$ time. A typical parser of this type is the MSTParser [McDonald et al., 2005b]. MSTParser use large-margin structured learning algorithms [McDonald et al., 2005a] to optimize the parameters of the models such that the score margin between the correct dependency graph and all incorrect dependency graphs are maximized.

The learning procedure is global since model parameters are set relative to the classification of the entire dependency graph, and not just over single arc attachment decisions. The main disadvantage of these models is that the feature representation is restricted to a limited number of graph arcs. This restriction is required so that both inference and learning are tractable.

5.2.2 Generic CYK Dependency Parsing

A projective dependency grammar can be thought of as a special case of the context-free syntax grammar where all terminal and non-terminals are in the word form. In other words, dependency grammar is equivalent to a fully lexicalized CFG where head-words are used to represent the syntactic category of their dominant constituents. Finding the optimal dependency parse of a sentence includes two related objectives:

1. Hierarchically segment the sentences into nesting constistuents, and

2. For each constituent, select one word as the head word.

Given this equivalence, we can modify the CYK parsing algorithm for the dependency grammar. Algorithm 1 shows the pseudo code for the parser. The key data structure in the algorithm is the parsing chart C. A cell $C[i, j]$ in the parsing chart corresponds to the span $e_i, \ldots, e_j$ of the input sentence. The chart is built bottom-up. The parser fills in cells of longer spans by combining partial hypotheses from shorter-spans. Each chart cell stores the best partial hypotheses for their corresponding span. For each cell the only factor that matters to the rest of the sentence is the choice of the headword. There is no need to keep those partial hypotheses with lower scores than the best one with the same head word. In other words, for each cell, the parser stores the best partial hypothesis for each possible head words for this span.

This generic CYK dependency parser parsing algorithm is based on the concept of *constituency*. For a sentence with n words, there are $O(n^2)$ substrings, each of which could be a constituent. The parser needs to find the optimal parse for each substring by considering $O(n)$ ways to combine hypotheses from two shorter substrings. For a substring of m words, there are m different parsing hypotheses each corresponds to an optimal parse with a distinct head word. m is of $O(n)$ and thus there are $O(n^2)$ ways shorter substring hypotheses can combine. In all, algorithm 1)is of $O(n^5)$ in time complexity.

```
   Data: Edge factor model $s$
   Input: Input: Sentence $e_1 e_2 \dots e_l$
 1 for $n \leftarrow 2$ to $l$ do
 2     for $i \leftarrow 1$ to $l - n + 1$ do
 3         for $j \leftarrow i$ to $i + n - 2$ do
 4             foreach partial hyp1 with head word $h_1$ in cell $[i, j]$ do
 5                 foreach partial hyp2 with head word $h_2$ in cell $[j + 1, i + n - 1]$ do
 6                     Create new $\text{hyp}_{12}$ with $h_1$ as the new head word;
                       $\text{score}(\text{hyp}_{12}) = \text{hyp}_1 + \text{hyp}_2 + s(h_1, h_2)$
 7                     Create new $\text{hyp}_{21}$ with $h_2$ as the new head word;
                       $\text{score}(\text{hyp}_{21}) = \text{hyp}_1 + \text{hyp}_2 + s(h_2, h_1)$
 8                     Add $\text{hyp}_{12}$ and $\text{hyp}_{21}$ to cell $[i, i + n - 1]$
 9                 end
10             end
11         end
12         Prune (if needed) partial hyps such that only the hyp with the highest score remains
           for each head word.
13     end
14 end
```

Algorithm 1: Edge-factored dependency parsing algorithm based on CYK.

5.2.3 Local, Greedy, Transition-based Parsing

A *transition system* for dependency parsing defines

1. a set C of *parser configurations*, each of which defines a (partially built) dependency graph G, and

2. a set T of *transitions*, each of which is a function $t : C \to C$, and

3. for every sentence $x = w_0, w_1, \dots, w_n$,

    (a) a unique *initial* configuration c_x

    (b) a set C_x of *terminal* configurations

A *transition sequence* $C_{x,m} = (c_x, c_1, \dots, c_m)$ for a sentence x is a sequence of configurations such that $c_m \in C_x$ and, for every $c_i (c_i \neq c_x)$, there is a transition $t \in T$ such that $c_i = t(c_{i-1})$. The dependency-graph assigned to x by $C_{x,m}$ is the graph G_m defined by the terminal configuration c_m.

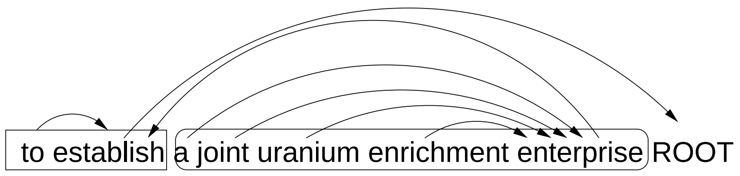


Figure 5-3: Span and constituents in dependency structure.

5.2.4 Cubic Time Dependency Parser

[Eisner, 2000] introduced a much faster parsing algorithm which is only $O(n^3)$ in time complexity. The key idea behind this cubic time dependency parsing algorithm is *parsing by span*. The generic dependency parser is based on the concept of *constituent*. A constituent is any substring consisting of a word and all its dependants. In other words, if e_i^j is a constituent and e_h is the root of this subtree, then for any $e_{k,k \neq h}$, $i \leq \mathbf{h}(e_k) \leq j$.

The concept of *span* is different from *constituent*. For span $[i, j]$, words e_i and e_j are called *end words* and $e_{i+1}, \ldots, e_{j-1}$ are the *interior words* of the span. A span must satisfy two conditions: 1) the head words of all interior words are inside the span; and 2) the only dependency links outside the span are through the end words. Formally, $[i, j]$ is a span if $i < k < j$, and e_k is a child or parent of $e_{k'}$, then $i \leq k' \leq j$. A span usually consists of two partial constituents. It can be a constituent when the root word of the subtree happens to be one of the end word of the span. In Figure 5-3, *to establish* is a valid span, but it is not a complete constituent since not all decedents of word *establish* are included. On the other hand, the span *a joint uranium enrichment enterprise* is both a valid span and a complete constituent.

When a dependency structure is projective, i.e. no crossing links exist, the structure can always be decomposed into nesting spans. A long span $[i, j]$ can be decomposed into two overlapping spans $[i, k]$ and $[k, j]$ $(i < k < j)$ in a deterministic procedure by identifying the rightmost word e_k that links to or from word e_i; if there is no such word, set $k = i + 1$. Because of projectivity, $[i, k]$ and $[k, j]$ must also be spans. Observing these properties, the dependency parsing can be viewed as combining overlapping spans to build longer spans till the complete sentence is covered. Since for each span, all interior words have already chosen their dependency parents inside the span and only the two end words matter to the rest of the sentence, the parser only needs to keep the optimal partial hypothesis for each end-word-status. In our implementation, there are only 3 valid end-word-status combination for each span. Combining two overlapping spans requires only $3 \times 3 = 9$, which is $O(1)$ time compared to $O(n^2)$ time in the generic CYK parser. This is the fundamental reason that we can reduce the time complexity from $O(n^5)$ to $O(n^3)$.

To implement the cubic-time dependency parser, we introduce a dummy word **ROOT** as the last word for each input sentence. To make sure that only one word in the actual sentence is linked to **ROOT** we use a flag variable *rootHasBeenChosen* to keep track of the root word selection.

Figure 5-4 shows the information for a partial hypothesis in the parse chart. b_1 is a boolean flag indicating whether e_i has chosen a parent inside the span. b_2 is the flat for e_j. There are 4 different combinations of b_1, b_2, namely,

- b_1=false, b_2=false: both e_i and e_j's parents are not inside the span.
- b_1=false, b_2=true: e_i does not have parent inside the span whereas e_j has a parent inside.
- b_1=true, b_2=false: e_i has a parent inside the span and e_j does not have parent inside.
- b_1=true, b_2=true: this should not happen, because it introduces circles inside the span.

Thus for each span, we only need to keep the best partial hypotheses for each of the three b_1, b_2 signatures. This also means that the parser only needs to consider at most $3 \times 3 = 9$ cases of combining two shorter spans into the longer span. Starting from initial partial hypotheses of two adjacent words (Figure 5-5), parser fills the parse chart bottom-up by applying the following operations on cell $[i, j]$ (Figure 5-6):

- Combine shorter spans: for all $k, i < k < j$, combine hypotheses in left span $[i, k]$ and hypotheses in right span $[k, j]$ if b_2 of left span is different from b_1 of the right span;
- Adding additional link from e_i to e_j on combined hypothesis if b_1 is false;
- Adding additional link from e_j to e_i on combined hypothesis if b_2 is false;
- Only keep the best hypothesis for each b_1, b_2 combination in each span.

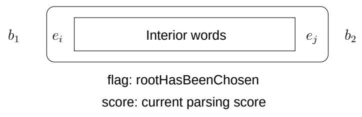


Figure 5-4: Span specification and its signatures in the parsing chart.

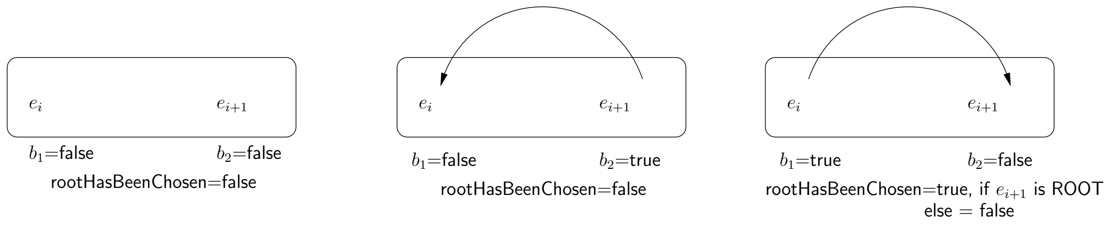


Figure 5-5: Seed hypotheses in the parsing chart.

The pseudo code of the cubic-time dependency parsing algorithm is described in Figure 5-7.

We compared the generic CYK dependency parser and the cubic-time parser on the same corpus of 1.6 M sentences. The dependency model contains probabilities of 330 million head/child word pairs. Table 5.1 shows the parsing time of the two parsers using the same model. Overall, the cubic-time parser is about 13 times faster than the generic CYK parser.

5.3 Modeling Sentence Structure With x-grams

The n-gram model is simple and robust. One of the reasons behind the success of the n-gram models is the Markov assumption. The Markov assumption states that each word in the sentence

	Generic CYK	Cubic-time
Parsing time	44,940 seconds	3,300 seconds
Avg. parsing time per sent.	0.02 second	0.002 second

Table 5.1: Comparing the generic CYK dependency parser and the cubic time parser on the same corpus of 1,619,943 sentences. Average sentence length is 23.2 words.

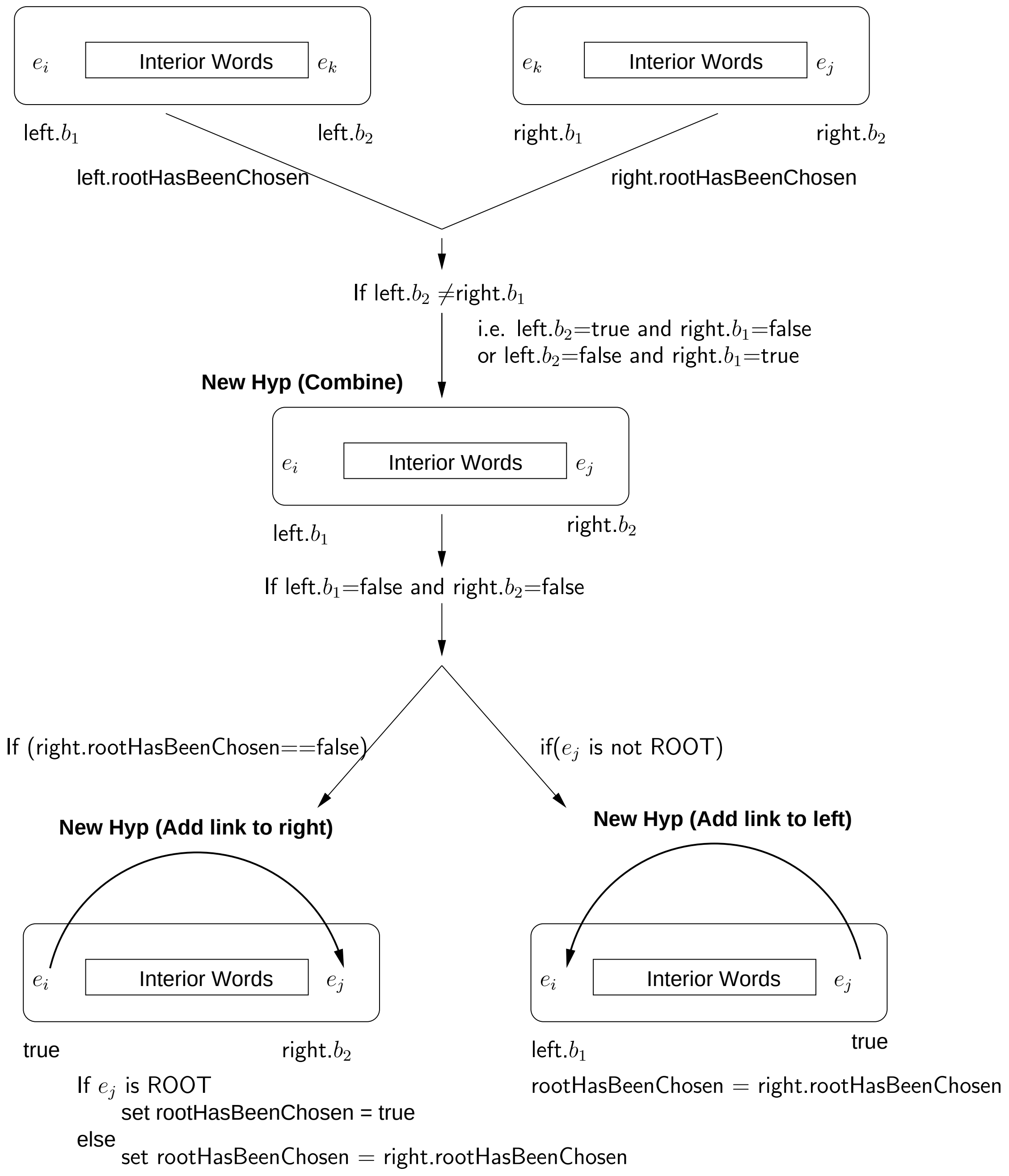

e_i
Interior Words
e_k
e_k
Interior Words
e_j
left.b_1
left.b_2
right.b_1
right.b_2
left.rootHasBeenChosen
right.rootHasBeenChosen
If left.b_2 $\neq$right.b_1
i.e. left.b_2=true and right.b_1=false
or left.b_2=false and right.b_1=true
New Hyp (Combine)
e_i
Interior Words
e_j
left.b_1
right.b_2
If left.b_1=false and right.b_2=false
If (right.rootHasBeenChosen==false)
if(e_j is not ROOT)
New Hyp (Add link to right)
New Hyp (Add link to left)
e_i
Interior Words
e_j
e_i
Interior Words
e_j
true
right.b_2
left.b_1
true
If e_j is ROOT
set rootHasBeenChosen = true
else
set rootHasBeenChosen = right.rootHasBeenChosen
rootHasBeenChosen = right.rootHasBeenChosen

Figure 5-6: Combine hypotheses from shorter spans.

```
Data: Edge factor model $s$
Input: Input: Sentence $e_1 e_2 \dots e_l$
Append Root to the end of input sentence;
for $i \leftarrow 1$ to $l$ do
    INSERT hypothesis [false, false] into span $[i, i+1]$;
    INSERT hypothesis [true, false] into span $[i, i+1]$;
    INSERT hypothesis [false, true] into span $[i, i+1]$;
end
for $n \leftarrow 3$ to $l+1$ do
    for $i \leftarrow 1$ to $l-n+2$ do
        $j \leftarrow i+n-1$;
        for $k \leftarrow i+1$ to $i+n-2$ do
            foreach LeftHyp in span $[i,k]$ do
                foreach RightHyp in span $[k,j]$ do
                    if LeftHyp.$b_2 \neq$ RightHyp.$b_1$ then
                        Combine LeftHyp and RightHyp to NewHyp1;
                        NewHyp1.rootHasBeenChosen $\leftarrow$ RightHyp.rootHasBeenChosen;
                        INSERT NewHyp1 into span $[i,j]$;
                        if NewHyp1.b1=false and NewHyp1.$b_2$=false then
                            if NewHyp1.rootHasBeenChosen=false then
                                ADD-LINK from $e_i$ to $e_j$ in NewHyp1, make it NewHyp2;
                                if $e_j$=ROOT then
                                    NewHyp2.rootHasBeenChosen = true;
                                end
                                INSERT NewHyp2 into span $[i,j]$
                            end
                            if $e_j \neq$ ROOT then
                                ADD-LINK from $e_j$ to $e_i$ in NewHyp1, make it NewHyp3;
                                INSERT NewHyp3 into span $[i,j]$;
                            end
                        end
                    end
                end
            end
        end
    end
end
```

Figure 5-7: Cubic-time dependency parsing algorithm based on span.

is conditionally independent of all earlier words given the immediate previous $n-1$ words.

$$P(w_i|w_1^{i-1}) = P(w_i|w_{i-n+1}^{i-1}) \tag{5.3}$$

This independence assumption makes it possible to estimate the probability of a sentence $\mathbf{e}$ over its "parts", the n-grams.

$$P(\mathbf{e}) = \prod_{i=1}^{|\mathbf{e}|} P(e_i|e_{i-n+1}^{i-1}) \tag{5.4}$$

Following this reasoning, we extend the idea of the n-gram model to the x-gram model. Assuming the structure of a sentence is given as a dependency tree, we can break the tree structure into structural "parts", the x-grams, which reflect the structure of the sentence.

Given the dependency parsing tree, we can break the tree in different ways.

5.3.1 d-gram Model

The d-gram model is based on the head-word chain structure. d words along the dependency path form a d-gram where each word is a modifier of its previous word. A $d2$-gram is a d-gram of order 2 which is a (head word, child word) pair. A $d3$-gram is a d-gram of order 3, formed by (head word of head word, head word, child word). In general, a dk-gram ($k = 2, 3, 4, \ldots$ is a sequence of k words along the dependency head-word chain, or in other words, d-gram of order k.

Table 5.2 shows all the d-grams of order 4 in the sentence. The first-order dependency parsing model is essentially a $d2$-gram model.

5.3.2 g-gram Model

The g-gram model is based on the hierarchical constituent structure. Each non-leaf node and all of its children ordered according to their positions in the sentence form a g-gram.

A g-gram corresponds to the skeleton of a constituent where the detailed information of all sub-constituents are shadowed by the head words. For each internal node i in a dependency tree we sort nodes i and its immediate children according to their positions in the sentence. Each resulting clause is the highest-level skeleton for the constituent rooted in node i. We call this clause the

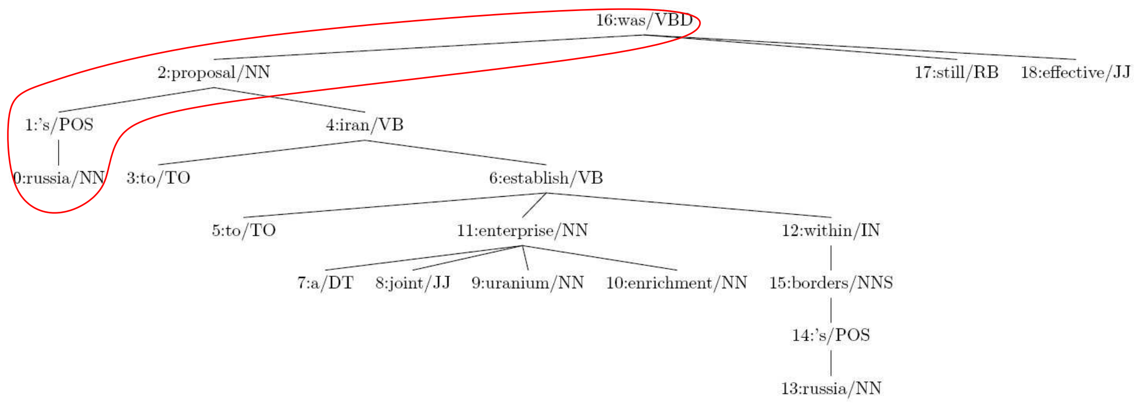


Figure 5-8: A *d*-gram:*was proposal 's russia* in sentence *russia's proposal to iran to establish a joint uranium enrichment enterprise within russia's borders was still effective*

was proposal 's russia
was proposal iran to
was proposal iran establish
proposal iran establish to
proposal iran establish enterprise
proposal iran establish within
iran establish enterprise a
iran establish enterprise joint
iran establish enterprise uranium
iran establish enterprise enrichment
establish with borders 's
within borders 's russia

Table 5.2: All *d*4-grams in sentence *russia's proposal to iran to establish a joint uranium enrichment enterprise within russia's borders was still effective*

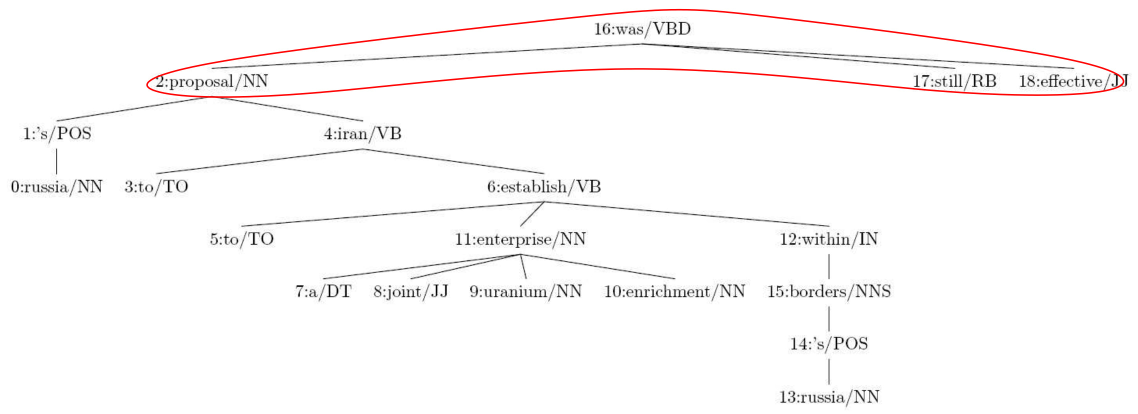


Figure 5-9: A g-gram:*proposal was still effective* in sentence *russia's proposal to iran to establish a joint uranium enrichment enterprise within russia's borders was still effective*

g-gram clause rooted in i. A k-th order g-gram is a sequence of k consecutive words in a g-gram clause. Figure 5-9 shows one g4-gram in sentence: *russia's* **proposal** *to iran to establish a joint uranium enrichment enterprise within russia's borders* **was still effective**. In g-gram *proposal was still effective*, modifiers of the head word *proposal* are ignored.

Table 5.3 lists all the g-grams in sentence *russia's proposal to iran to establish a joint uranium enrichment enterprise within russia's borders was still effective*. From the perspective of the sentence string, g-gram is a *gapped* n-gram, or a skipped n-gram where the skipped words are based

proposal was still effective
's proposal iran
russia 's
to iran establish
to establish enterprise within
a joint uranium enrichment enterprise
within borders
's borders
russia 's

Table 5.3: All g-grams in sentence *russia's proposal to iran to establish a joint uranium enrichment enterprise within russia's borders was still effective*

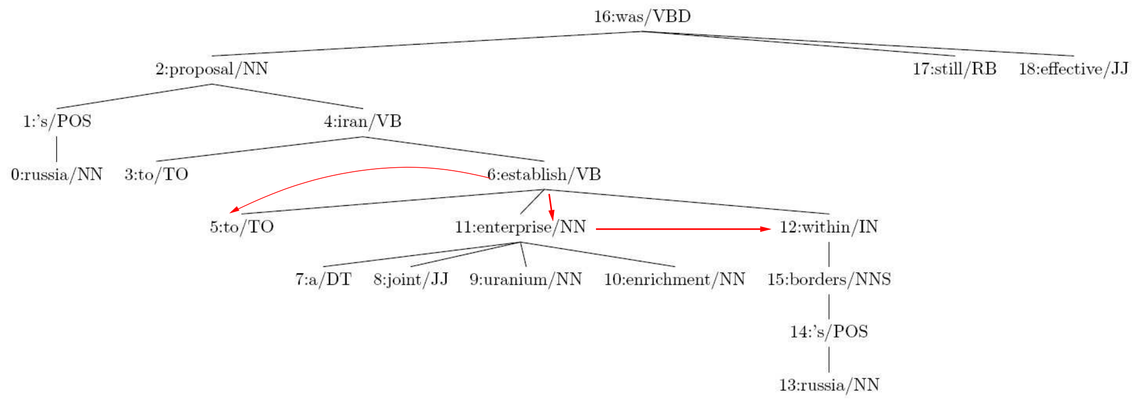


Figure 5-10: h-gram: *establish to* and *establish within* as generated from the head word *establish*.

on the structure of the sentence.

5.3.3 h-gram Model

Similar to the g-gram model, the h-gram model is also based on the hierarchical constituent structure. Instead of ordering words according to their positions in the sentence string from left to right, h-gram model orders words centered at the head word and expands to left and right. In other words, a constituent is generated in three steps: generating the head word; generating all the words left of the head word; generating all the words positioned right of the head word. Starting from the head word, adding all the children nodes that are left to the head forms a h-gram left-direction clause. Similarly we can create the h-gram right direction clause. The k-th order left-direction h-gram is a sequence of k words in the h-gram left-direction clause, and similarly for the k-th order right-direction h-gram.

Figure 5-10 illustrates the process of creating the constituent rooted at node *establish*. Based on this head word, we add *to* to its left and *enterprise* to its right. Conditioned on the head-centered context *establish enterprise*, we can further add word *within* into this constituent. This process resulted in h-grams: *establish to* and *establish enterprise within*. h-gram differs from g-gram mainly on words that are left to the head word. Words in g-gram are always generated from left to right

was proposal
was still effective
still effective
proposal 's
proposal iran
's russia
iran to
iran establish
establish to
establish enterprise within
enterprise enrichment uranium joint a

Table 5.4: All h-grams in sentence *russia's proposal to iran to establish a joint uranium enrichment enterprise within russia's borders was still effective*

where as in h-gram words are generated from the center (headword).

h-gram models are usually used in dependency parsers.

5.3.4 x-gram Edge Length

We use the x-grams to model and reflect the structure in a sentence, especially to capture the long-distance dependencies. One interesting statistics is the average edge length of $x2$-grams. The average $x2$-gram edge length indicates if x-grams really capture longer distance dependencies than n-grams since the $n2$-gram edge length is always 1. Table 5.5 shows the averaged $d2$ and $g2$-gram edge length from the dependency trees of the Gigaword data. Table 5.6 compares the averaged $d2$ and $g2$-gram edge length of the human reference translation and MT output for the NIST MT03 Arabic-English evaluation set.

The averaged $d2$-gram and $g2$-gram edge length vary for different data set, but it is clear that x-gram captures more long distance dependencies than the n-gram models.

Corpus	Avg. $d2$-gram edge len.	Avg. $g2$-gram edge len.
APW	3.228	2.052
AFP	2.966	2.140
FBIS	2.992	2.120
LTW	2.845	1.982
NYT	3.196	2.058
WSJ	2.968	2.074
XIN	2.995	2.148

Table 5.5: Average d-gram and g-gram edge length calculated over different data.

Data		Avg. $d2$-gram distance	$g2$-gram distance
Human Reference	ahd	3.275	2.319
	ahe	3.333	2.388
	ahg	3.391	2.329
	ahi	3.385	2.385
COTS MT	ame	3.304	2.422
Research MT	ara	3.159	2.297
	arb	3.477	2.309
	ari	3.124	2.315
	arm	3.209	2.340
	arp	3.086	2.341

Table 5.6: Averaged $d2$-gram and $g2$-gram edge length in different corpora. Edge length is the distance of the two words in the sentence.

5.4 x-gram Language Model

The probability of a sentence $P(\mathbf{e})$ can be estimated as:

$$P(\mathbf{e}) = \sum_{\pi} P(\mathbf{e}, \pi) \tag{5.5}$$

where π is a possible parse of $\mathbf{e}$. Assuming that the joint probability $P(\mathbf{e}, \pi)$ of using the optimal parse π^* is much larger than all other alternative parses, $P(\mathbf{e})$ can be approximated by:

$$\begin{aligned} P(\mathbf{e}) &\approx P(\mathbf{e}, \pi^*) && (5.6) \\ &= P(\mathbf{e}|\pi^*)P(\pi^*) && (5.7) \end{aligned}$$

We can further assume that $P(\pi^*) = 1$, or in other words, there is only one parse of the sentence, then

$$\begin{aligned} P(\mathbf{e}) &\approx P(\mathbf{e}|\pi^*) && (5.8) \\ &= \prod_{i}^{|\mathbf{e}|} P(e_i|\pi^*) && (5.9) \end{aligned}$$

Applying the Markov independency assumption on Equation. 5.9

$$P(e_i|\pi^*) = P(e_i|\pi^*_x(i)) \tag{5.10}$$

where $\pi^*_x(i)$ is the corresponding x-gram structural context for word e_i in the sentence.

5.4.1 n-gram Language Model

n-gram language model can be considered as a special case of the structural x-gram language model where the structure π is a chain such as the one illustrated in Figure 7-5.

The context of word e_i in the n-gram model is a sequence of $n-1$ words: e_{i-n+1}^{i-1}.

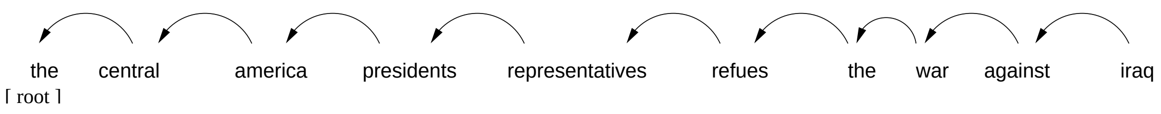


Figure 5-11: Bootstrap the structure of a sentence as a Markov chain.

5.4.2 d-gram Language Model

For d-gram language models, the structural context of a word e_i is the chain of its $d-1$ ancestors in the dependency tree, namely: $e_{\mathbf{h}^{d-1}(i)}, e_{\mathbf{h}^{d-2}(i)}, \cdots, e_{\mathbf{h}(i)}$.

Training d-gram Language Model

The training of the the d-gram language model is based on the frequency statistics of d-gram types in a parsed corpus. This is similar to the training of the n-gram language model which starts by counting the n-gram frequencies. However, one major difference is n-gram counts satisfy the constrain that:

$$C(w_1^{n-1}) = \sum_{w_n} C(w_1^{n-1} w_n) \tag{5.11}$$

which leads to:

$$\sum_{w_n} P(w_n | w_1^{n-1}) = 1 \tag{5.12}$$

But this is not the case for d-gram counts. For example in Figure 5-8 *proposal iran* occurs ones as a $d2$-gram, but it occurs twice as the lower-order context for $d3$-gram *proposal iran to* and *proposal iran establish*. $C(\textit{proposal iran}) = 1$ and $\sum_{\bullet} C(\textit{proposal iran}\bullet) = 2$ obviously violates the constraint. We need to use a different statistics other than the raw counts to satisfy eq. 5.12.

Denote the number of leaf nodes dominated by node i in a particular dependency tree as $F(i)$. $F(i)$ can be recursively defined as:

$$F(i) = \begin{cases} \sum_{j \in \{v : \mathbf{h}(v) = i\}} F(j) & \\ 1 & \text{node i is leaf node} \end{cases} \tag{5.13}$$

Figure 5-12 shows the $F(i)$ value for each node in a dependency tree.

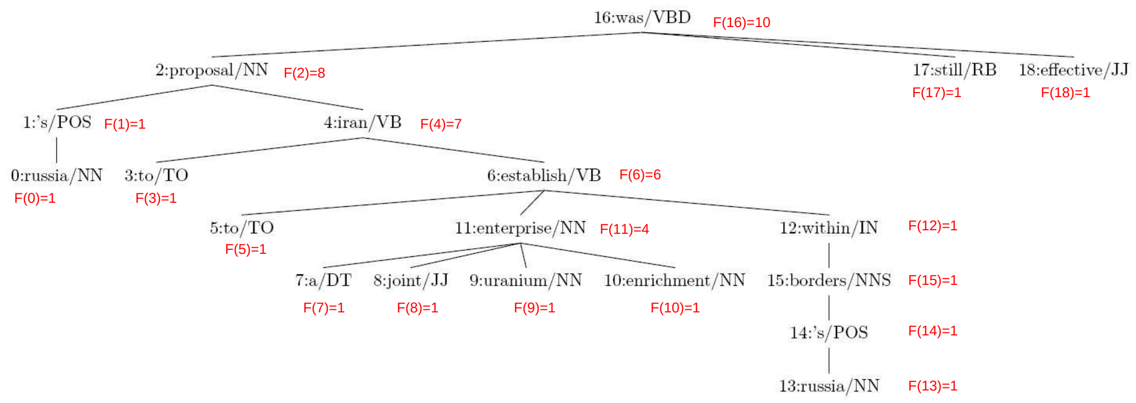


Figure 5-12: Number of dominated leaf nodes for each node in the tree. Showing next to each node in red font are the $F(i)$ values.

By definition, $F(i)$ is the sum over all node i's children's F values. If we use $F(i)$ as the statistics for d-grams ending at word e_i we can show that over the whole training data

$$F(w_1^{n-1}) = \sum_{\bullet} F(w_1^{n-1})\bullet \tag{5.14}$$

For the example, from a single tree shown in Figure 5-8, $F(\textit{proposal iran}) = 7$, $F(\textit{proposal iran to}) = 1$ and $F(\textit{proposal iran establish}) = 6$.

From all the parsed trees in the training corpus, we can collect the F-frequencies for all d-gram types up to a certain order. Treat d-grams as n-grams and F-frequencies as counts, we can use the well-established n-gram smoothing techniques such as the Modified Kneser-Ney smoothing [Kneser and Ney, 1995] to estimate the conditional probability of a word given its d-gram context.

It is worth mentioning that the dependency parser model is fundamentally a $d2$-gram model.

Estimating $P(\mathbf{e})$ Using d-gram LM

Given the dependency parse π of a testing sentence $\mathbf{e}$, we can estimate $P(\mathbf{e})$ using the trained d-gram model as:

$$\begin{aligned} P(\mathbf{e}|\pi) &= \prod_{i}^{|e_i|} P(e_i|\pi) && (5.15) \\ &= \prod_{i}^{|e_i|} P(e_i|\pi_d(i)) && (5.16) \\ &= \prod_{i}^{|e_i|} P(e_i|e_{\mathbf{h}^{d-1}(i)}, e_{\mathbf{h}^{d-2}(i)}, \cdots, e_{\mathbf{h}(i)}) && (5.17) \end{aligned}$$

Similar to the n-gram language model, backoff is used when the higher order d-gram does not exist in the trained d-gram model.

5.4.3 g-gram Language Model

The training of the g-gram language model starts with the process of reading off g-grams from the parsed trees. For each parse tree in the training corpus, we can read out all the g-grams such as shown in table 5.3. Counts of g-gram types of different order are added over all parse trees in the training set. g-grams are essentially gapped n-gram, its training is the same as the n-gram model.

Estimating $P(\mathbf{e})$ Using g-gram LM

Each non-terminal node in the dependency tree appears in two different g-gram contexts: one as a child node and another as a parent. For example, word *proposal* in dependency tree fig. 5-9 is a child node of *was* in g-gram *proposal was still effective*. It also appears in g-gram *'s proposal iran*. Thus each non-terminal node has two structural contexts in the g-gram model to estimate $P(e_i|\pi)$. In this example, word *proposal* can be predicted by the constituent start symbol $< c >$ if we choose g-gram *proposal was still effective* as its context, or it can be $< c >' s$ if we consider g-gram *'s proposal iran* to be the context. In this thesis, we choose the structural g-gram context that gives the highest probability for each word to estimate the word probability.

word	g-gram	n-gram
russia	-3.69	-2.86
’s	-0.62	-0.81
proposal	-2.78	-2.10
to	-2.63	-0.60
iran	-3.22	-2.36
to	-2.63	-1.53
establish	-2.56	-2.57
a	-3.22	-0.43
joint	-2.20	-1.30
uranium	-3.07	-3.07
enrichment	-0.03	-0.03
enterprise	-3.49	-5.12
within	-3.35	-3.48
russia	-3.69	-3.10
’s	-0.62	-0.96
borders	-2.24	-2.86
was	**-0.69**	**-2.15**
still	-2.18	-2.36
effective	-3.71	-3.73
$< s >$		-3.06
sum	-46.7181	-44.60

Table 5.7: $\log P(e_i|\pi)$ estimated by the g-gram model and the n-gram language model.

5.4.4 h-gram Language Model

The generative process under the h-gram model starts by generating the head word of a constituent first. Based on the chosen head word we then expand to the left and right to complete the constituent. Repeat this process for all the constituents in the sentence till all words are generated. For sentence shown in Figure 5-10, we first need to generate word *was* with probability $P(\text{was}|\text{-root-})$. Then based on the headword *was*, generate *proposal* with probability $P(\text{proposal}|\text{was})$ to the left and generate *still* with $P(\text{still}|\text{was})$ and *effective* with $P(\text{effective}|\text{was still})$ to the right direction. Repeat this process on constituents rooted on word *proposal* till all words are created.

Training h-gram Language Model

The h-gram language model needs to model three types of probabilities inline with the generative process: first, the probability of a word being the root node of a sentence; the probability of a word as the left context given the head word and other siblings to its right; and the probability of a word as the right context of the head word and other siblings to its left. Corresponding h-grams can be read off from the parsed training corpus and the head-to-left and head-to-right models can be trained separately based on the counts of h-grams of corresponding directions.

Estimating $P(\mathbf{e})$ Using h-gram LM

Given the dependency parsing tree π of a sentence $\mathbf{e}$, the probability of e_i depends on its location with regard to the head word. If e_i is the root node of the whole tree, $P(e_i|\pi) = P(e_i|\text{-root-})$; if e_i is on the left side of the head word, $P(e_i|\pi) = P(e_i|\text{right siblings} \cdots \text{head word})$; if e_i is on the right side of the head word, $P(e_i|\pi) = P(e_i|\text{left siblings} \cdots \text{head word})$;

5.4.5 Dynamic Structural Context x-gram Language Model

n-gram, d-gram, g-gram and h-grams have different underlining assumptions of which structural context best predicts a word in the sentence. Natural language is very complicated and no single type of structural context can account for the generation of all words in a sentence. For sentence shown in Figure 5-1, word *was* is best predicted by its g-gram context *proposal* whereas word *borders* probably depends more on its d-gram context *establish within*.

We are looking for a combination function f which combines the probabilities of a word from different structural contexts:

$$P(e_i|\pi) = f(P(e_i|\pi_n(i)), P(e_i|\pi_d(i)), P(e_i|\pi_g(i)), P(e_i|\pi_h(i), \cdots)) \tag{5.18}$$

There are several ways to estimate the probability of a word given different structural contexts such as linear interpolation, log-linear model, etc. In this thesis, we choose max for the combination function f. In other words, for each word e_i, we use the structural context that best predicts e_i to

	n-gram	d-gram	g-gram	h-gram
Hyp.	36.0%	26.6%	21.0%	16.5%
Ref.	34.5%	28.7%	20.5%	16.3%

Table 5.8: Structural context that best predicts a word.

estimate $P(e_i|\pi)$.

Table 5.8 shows the percentage of different structural context best predicts a word in human and MT generated sentences. The testing data is the NIST 03 Arabic/English set. n-gram, d-gram, g-gram and h-gram models are trained from the same 500M word corpus and its dependency parsing trees up to order 3 and smoothed by the modified Kneser-Ney method. Not surprisingly, n-gram context is most likely useful in predicting the next word in the sentence. This explains in part why the n-gram language model has good performances even though it does not capture long-distance dependencies. On the other hand, structured context better predicts a word than the n-gram model in other cases. This supports our argument that structured context is important in language modeling.

5.5 Related Work

There has been much effort recently in MT on adding syntactically motivated features. Och and others [Och et al., 2004] investigated the efficacy of integrating syntactic structures into a state-of-the-art SMT system by introducing feature functions representing syntactic information and discriminatively training scaling factors on a set of development N-best lists. They obtained consistent and significant improvement from the implicit syntactic features produced by IBM model 1 scores, but rather small improvement from other syntactic features, ranging from shallow to deep parsing approaches. Recently, Hasan and others [Hasan et al., 2006] observed promising improvement of MT performance in a reranking framework by using supertagging and lightweight dependency analysis, a link grammar parser, and a maximum entropy based chunk parser. They achieved up to 0.7% absolute increase on BLEU on C-Star 03 and IWSLT 04 tasks. [Wang et al., 2007] investigate the use of linguistically motivated and computationally efficient structured language models for reranking N-best hypotheses in a statistical machine translation system. Small gains (about 0.1% to 0.7% absolute BLEU) are observed on some genres while on some other genres BLEU scores drop slightly observed on the blind test set

[Collins and Duffy, 2001] uses tree kernel in parsing. The kernel functions are instances of "convolution kernels", which involves a recursive calculation over "parts" of a discrete structure. The key idea is to take a structured object and split it up into parts. If one can construct kernels over the parts then one can combine these into a kernel over the whole objects. The recursive combination of the kernels over parts of an object retains information regarding the structure of that object.

Chapter 6

Structure Induction

Statistical parsers induce the parsing models from human labeled data: tree-banks, such as the Penn-treebank [Marcus et al., 1993]. Treebanks require huge amount of human effort to construct. As a result, treebanks are usually small in size, limited in domain and only available for a couple of high density languages.

On the other hand, there exists vast amount of text in the electronic forms especially after Internet becomes popular. This is a typical problem in statistical learning. Semi-supervised and unsupervised methods are important because good labeled data is expensive, whereas there is no shortage of unlabeled data. A simple method of incorporating unlabeled data into a new model is self-training. In self-training, the existing model first labels unlabeled data. The newly labeled data is then treated as truth and combined with the actual labeled data to train a new model. This process can be iterated over different sets of unlabeled data if desired.

Language	Words	Domain	Source
Arabic	300K (1M planned)	News	AFP newswire
Chinese	0.5M	Broadcasting news	Xinhua newswire, Sinorama and HK news
English	4.5M	Financial news	Wall Street Journal
Korean	54K	Military	Military language training manuals

Table 6.1: Major treebanks: data size and domain

In this thesis, we use self-training to induce structures for unlabelled data and use the induced structure to train the structured langauge model. The unlabeled data is more relevant in domain with the testing data compared to the treebank. The structured langauge model trained on the induced structure of the unlabeled data is then more relevant to the testing domain.

6.1 Semisupervised Learning Through Self-training

Self-training is one of the earliest semi-supervised learning algorithms. Self-training assumes that the existing model is accurate enough to label the unlabeled data. The newly labeled data is then treated as truth and combined with the actual labeled data to train a new model. This process can be iterated over different sets of unlabeled data if desired.

Self-training is one of the simplest semi-supervised learning method. It is also a wrapper method which can be applied to existing learning algorithms to make use of the unlabeled data. There are two disadvantages with self-training approaches. Firstly, since we assume that labels from the current model are always correct, early mistakes could reinforce themselves in the following iterations. This is the case even if the labeled data is associated with confidence calculated by the current model. Secondly, it is generally not clear whether updated model converges.

Despite these obvious disadvantages, self-training has been used in a variety of applications to make use of the unlabeled data. Ng and Cardie, (2003b) implement self-training by bagging and majority voting. A committee of classifiers are trained on the labeled examples, then classify the unlabeled examples independently. Only those examples, to which all the classifiers give the same label, are added to the training set and those classifiers are retrained. This procedure repeats until a stop condition is met. Clark et al., (2003) uses self-training to retrain a tagger on its own labeled cache on each around.

In this thesis, we use self-training to update the structure of the unlabeled data for structured language model training (Figure 6-1). We first train a dependency parser from 44K human labeled parse trees. We call this parser Treebank DepParser. The unlabeled data is parsed using the Treebank DepParser to bootstrap the initial structure. Assuming that the parse is of high confidence, we train the x-gram language model from the bootstrapped data. Given the x-gram language

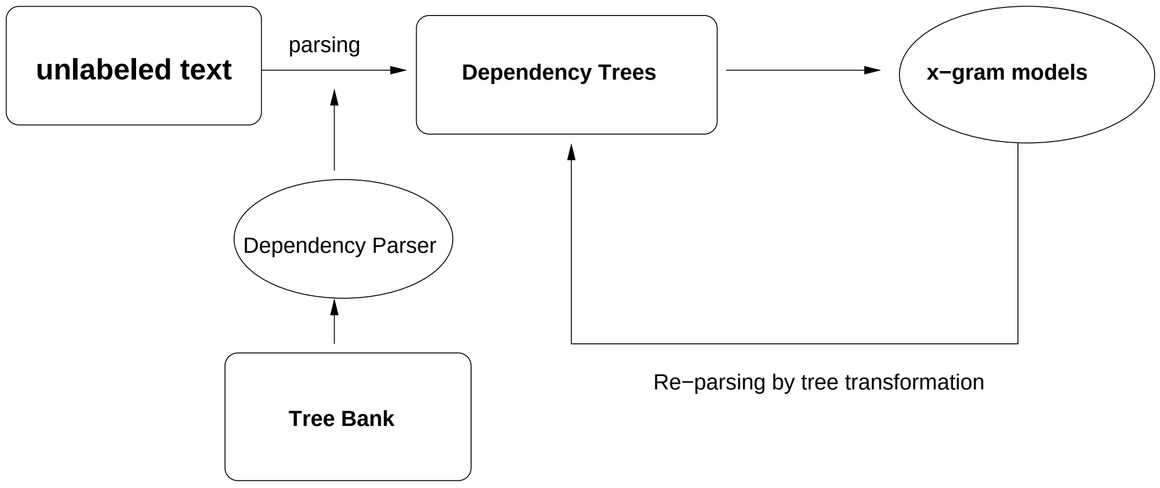


Figure 6-1: Self-training the structured language model through tree-transformation.

model, bootstrapped trees for the unlabeled data are transformed to maximize the probability of each sentence. The transformed trees are now having new structures which we assume are better than the bootstrapped structure. Repeat the above process till stopping condition is satisfied.

One of the key step in this process is the dependency tree transformation.

6.1.1 Dependency Tree Transformation

We propose three tree operations to transform a tree to a different structure. For a word w_i in the dependency tree, we could:

- MoveSubTreeUp: move the subtree rooted at node w_i up in the tree (Figure 6-2);
- MoveSubTreeDown: make w_i and as one of its sibling's child (Figure 6-3);
- SwapWithParent: make w_i as the new parent of all its siblings and make parent of w_i now w_i's direct child. (Figure 6-4);

operation *MoveSubTreeUp* and *MoveSubTreeDown* moves the subtree up and down in the structure whereas *SwapWithParent* rotates a subtree by choosing a new word as the head of this constituent.

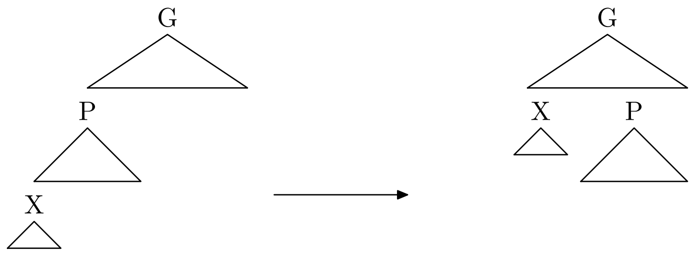


Figure 6-2: Tree transformation: move sub tree X up as parent's sibling

Figure 6-3: Tree transformation: move sub tree X down as its sibling Y's child.

Applying one of the above tree transformations on a dependency tree results in a new tree. With one tree transformation, one dependency tree can be transformed into three different new trees. There are in total $O(3^N)$ possible transformed trees after applying N tree transformations. For a given tree evaluation metric, we want to improve the metric score for a particular sentence by transforming its initial dependency tree, which is equivalent to search in the $O(3^N)$ different trees the one with the highest metric scores. Because the search space is huge, we use the greedy search to find the approximate optimal best trees. All transformed dependency trees are stored in a priority queue of *beam size* according to their metric scores. At each step, the top tree is popped out of the queue and tree transformation is applied on the current best tree. As the result of the tree transformation, three transformed trees are inserted into the priority queue according to their metric scores. To keep the priority queue in a reasonable size, we keep only up to *beamsize* transformed trees in the queue. Tree transformation algorithm ends when transformed trees can not improve the metric scores any more.

Figure 6-4: Tree transformation: swap X with its parent P.

In our experiments, we apply tree transformation both on dependency trees generated by the MST dependency parser and also on the bootstrapped chain structure for each sentence. In the latter case, we bootstrap the dependency structure for a sentence as a chain when parsers are not available and apply a sequence of tree transformations to transform the chain structure into more meaningful trees to maximize some tree evaluation metrics.

Figure 6-5 to 6-17 show the dependency tree generated by the MST parser and various trees transformed from either the parser output or the chain structure to maximize different tree evaluation metrics.

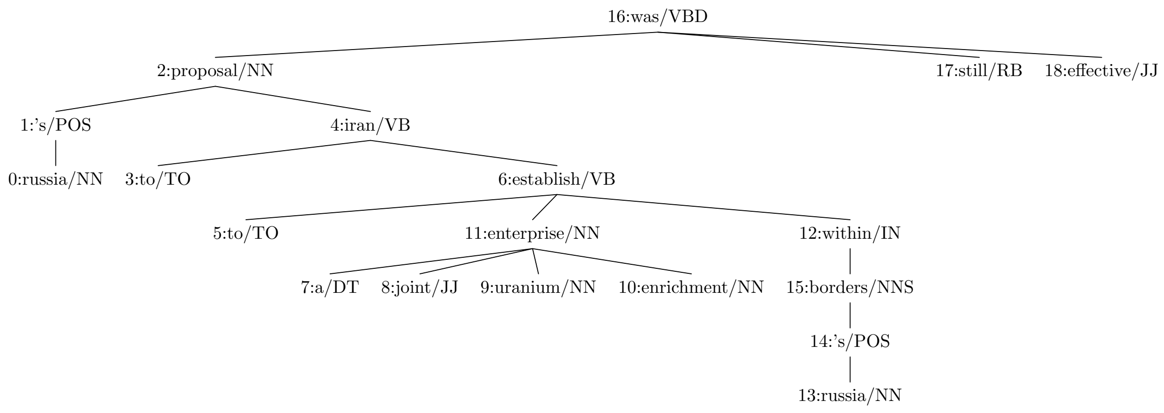


Figure 6-5: Dependency tree from the parser.

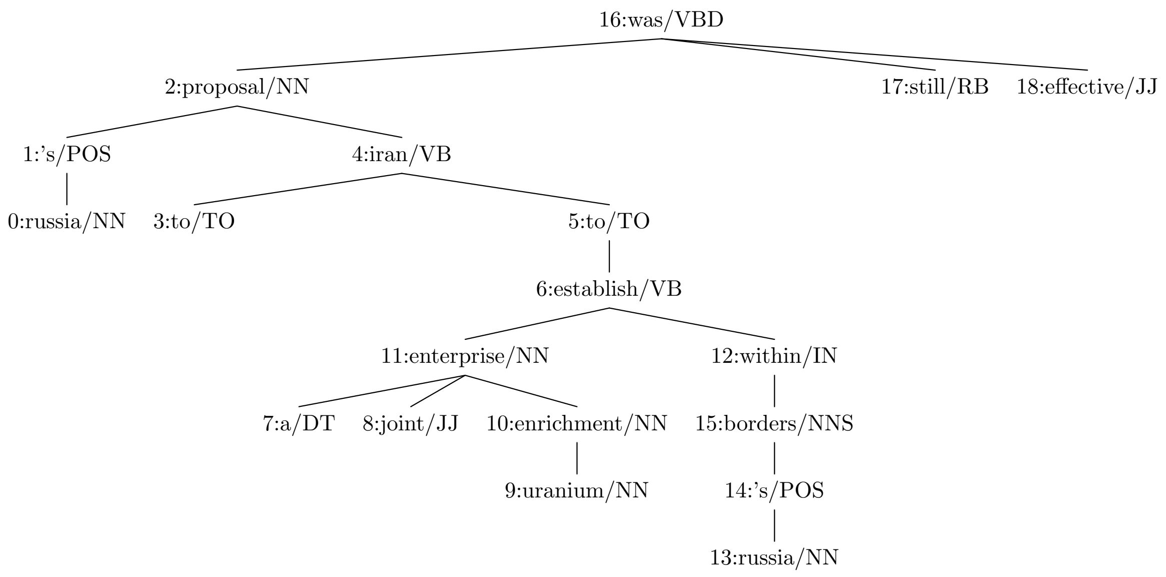


Figure 6-6: Transformed dependency tree with maximum *d*-gram probability. Transformed from the parser output.

6.2 Unsupervised Structure Induction

In the case where there is no labeled data, inducing structure becomes an unsupervised learning problem. Learning structure of natural language from pure text is an ambitious task in NLP and machine learning research. Various methods have been tried with moderate success in unsupervised structure induction.

There are two key issues with unsupervised structure induction: what is the desired structure and how to evaluate the usefulness of the induced structure. For the language modeling task, the objective of structure induction is to better predict words given the structure of the sentence. In other words, we are hoping that more long distance dependency patterns can be captured through the structures in language.

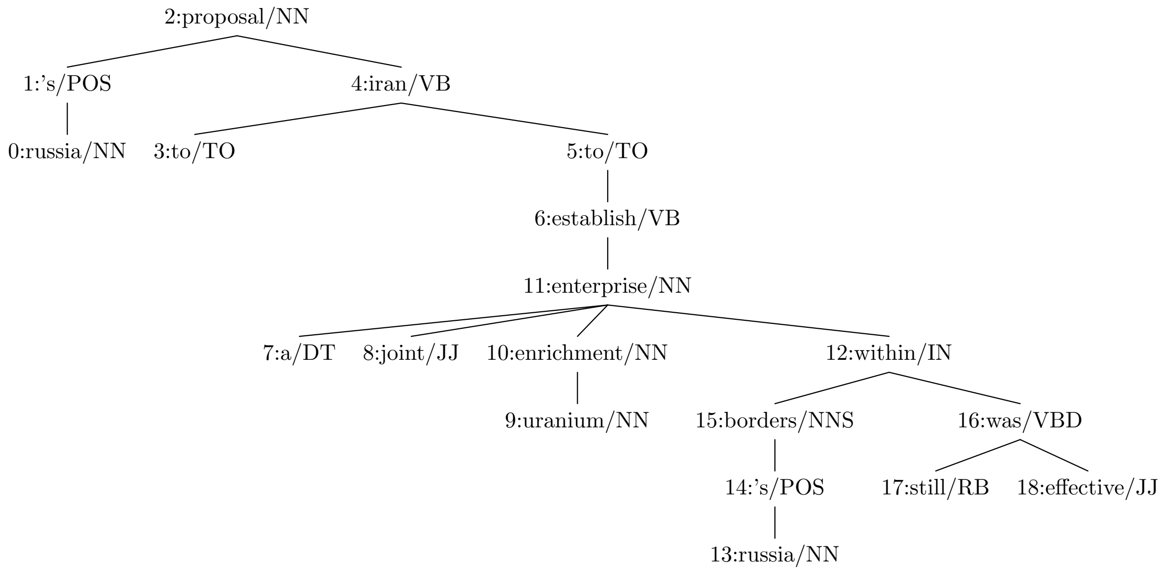


Figure 6-7: Transformed dependency tree with maximum *d*-gram probability. Transformed from the chain structure.

6.2.1 Model 1

We propose a unsupervised dependency model: model 1[1]. Denote the joint probability of a sentence $\mathbf{e}$ and a particular dependency parse π as $P(\mathbf{e}, \pi)$. It can be estimated as a product of the probabilities of each word e_i conditioned on its head word $\mathbf{h}_\pi(e_i)$, where $\mathbf{h}_\pi(\cdot)$ is the head-word mapping function specified by parse π. The probability of a sentence $P(\mathbf{e})$ is then the joint probabilities summed over all possible parses.

$$P(\mathbf{e}) = \sum_{\pi} P(\mathbf{e}, \pi) \tag{6.1}$$

Assuming that all possible structures of a sentence are of the same probability $1/(l+1)^l$, then

$$P(\mathbf{e}, \pi) = \frac{1}{(l+1)^l} \sum_{\pi} \prod_{i} P(e_i | \mathbf{h}_\pi(e_i)). \tag{6.2}$$

[1]As the name suggests, the model and its derivation is inspired by the IBM translation model1 which uses EM algorithm to induce the source/target word to word translation probabilities.

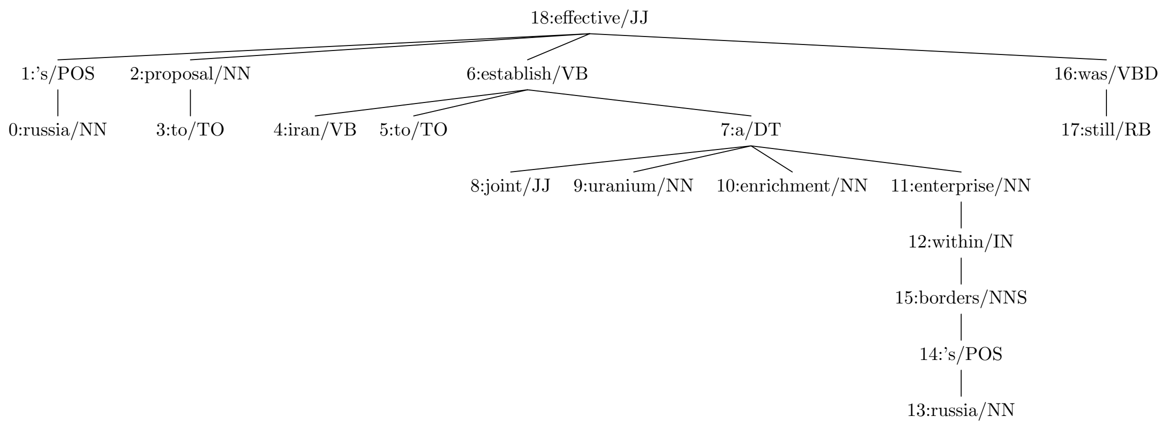


Figure 6-8: Transformed dependency tree with maximum g-gram probability. Transformed from the parser output.

The dependency tree is determined by specifying the values of $\mathbf{h}_\pi(e_i)$ for i from 1 to $l = |\mathbf{e}|$. $\mathbf{h}_\pi(e_i) = 0$ if e_i is the root of the dependency tree. There should be only one root word in a valid dependency tree. To make the derivation easier, we will be slack on this restriction and $\mathbf{h}_\pi(e_i)$ can take any value from 0 to l. In other words, word e_i can choose any word in the sentence as its headword.[2]

Therefore,

$$P(\mathbf{e}) = \frac{1}{(l+1)^l} \sum_{\mathbf{h}_\pi(e_1)=0}^{l} \sum_{\mathbf{h}_\pi(e_2)=0}^{l} \cdots \sum_{\mathbf{h}_\pi(e_l)=0}^{l} \prod_{i=1}^{l} P(e_i|\mathbf{h}_\pi(e_i)) \tag{6.3}$$

We wish to adjust the $d2$-gram probabilities so as to maximize $P(\mathbf{e})$ subject to the constraints that for each head word h[3],

$$\sum_e P(e|h) = 1 \tag{6.4}$$

We want to optimize the probabilities of $P(e|h)$ to maximize the probability of each sentence in the training corpus under the constraint of equation 6.4. Introduce Lagrange multipliers λ_h and

[2] A word can not be headword of itself, this can be easily addressed in the implementation. The derivation is more clear when $\mathbf{h}_\pi(e_i)$ is alllowed to take any value from 0 to l includiing i itself.

[3] We use h to denote the headword even though h and e come from the same vocabulary.

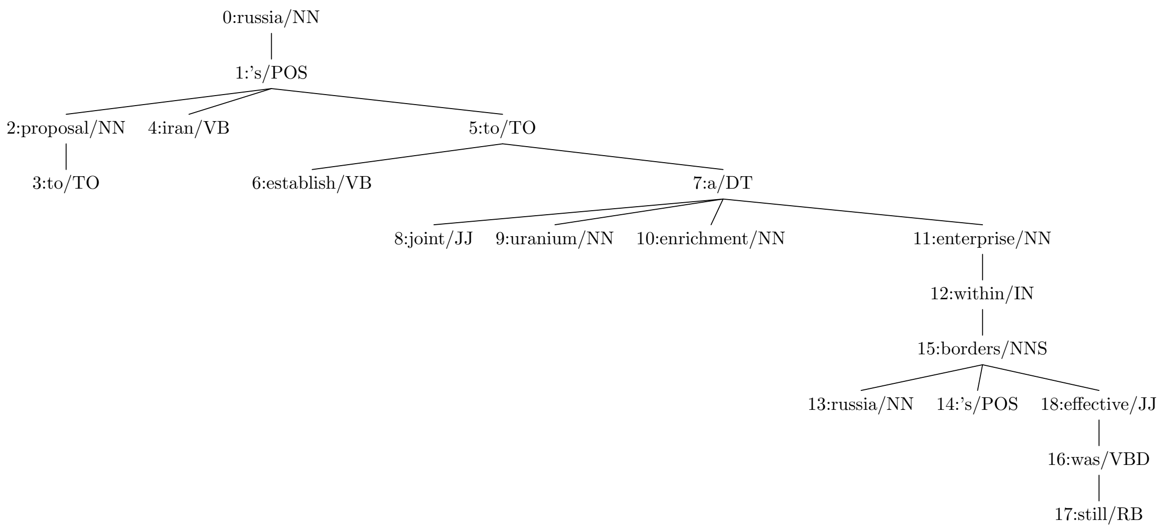


Figure 6-9: Transformed dependency tree with maximum g-gram probability. Transformed from the chain structure.

the auxiliary function

$$\Lambda(P,\lambda) \equiv \frac{1}{(l+1)^l} \sum_{\mathbf{h}_\pi(e_1)=0}^{l} \sum_{\mathbf{h}_\pi(e_2)=0}^{l} \cdots \sum_{\mathbf{h}_\pi(e_l)=0}^{l} \prod_{i=1}^{l} P(e_i|\mathbf{h}_\pi(e_i)) - \sum_h \lambda_h (\sum_h P(e|h) - 1) \tag{6.5}$$

The partial derivative of Λ with respect to $P(e|h)$ is:

$$\frac{\partial \Lambda}{\partial P(e|h)} = \frac{1}{(l+1)^l} \sum_{\mathbf{h}_\pi(e_1)=0}^{l} \sum_{\mathbf{h}_\pi(e_2)=0}^{l} \cdots \sum_{\mathbf{h}_\pi(e_l)=0}^{l} \sum_{i=1}^{l} \delta(e,e_i)\delta(h,\mathbf{h}_\pi(e_i))P(e|h)^{-1} \prod_{j=1}^{l} P(e_j|\mathbf{h}_\pi(e_j)) - \lambda_h \tag{6.6}$$

where δ is the Kronecker delta function

$$\delta(x,y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{if } x \neq y \end{cases} \tag{6.7}$$

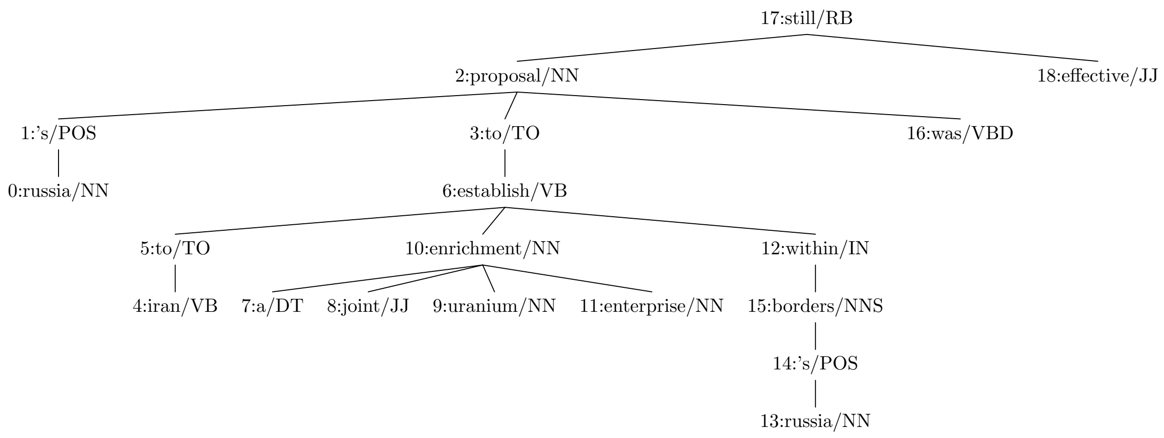


Figure 6-10: Transformed dependency tree with maximum h-gram probability. Transformed from the parser output.

Let $\frac{\partial \Lambda}{\partial P(e|h)} = 0$,

$$P(e|h) = \lambda_h^{-1} \frac{1}{(l+1)^l} \sum_{\mathbf{h}_\pi(e_1)=0}^{l} \sum_{\mathbf{h}_\pi(e_2)=0}^{l} \cdots \sum_{\mathbf{h}_\pi(e_l)=0}^{l} \sum_{i=1}^{l} \delta(e, e_i)\delta(h, \mathbf{h}_\pi(e_i)) \prod_{j=1}^{l} P(e_j|\mathbf{h}_\pi(e_j)) \qquad (6.8)$$

Equation 6.8 suggests using the EM algorithm to iteratively update the $P(e|h)$. From each iteration, we can estimate the expected number of times that h is the head word of e in a sentence , defined as

$$c(e|h; \mathbf{e}) = \sum_{\pi} P(\mathbf{e}, \pi) \sum_{i=1}^{l} \delta(e, e_i)\delta(h, \mathbf{h}_\pi(e_i)), \qquad (6.9)$$

Use equation 6.2 to rewrite equation 6.8, we have

$$P(e|h) = \lambda_h^{-1} \sum_{\pi} P(\pi|\mathbf{e}) \sum_{i=1}^{l} \delta(e, e_i)\delta(h, \mathbf{h}_\pi(e_i)) \qquad (6.10)$$

where $P(\pi|\mathbf{e}) = P(\mathbf{e}, \pi)/P(\mathbf{e})$.

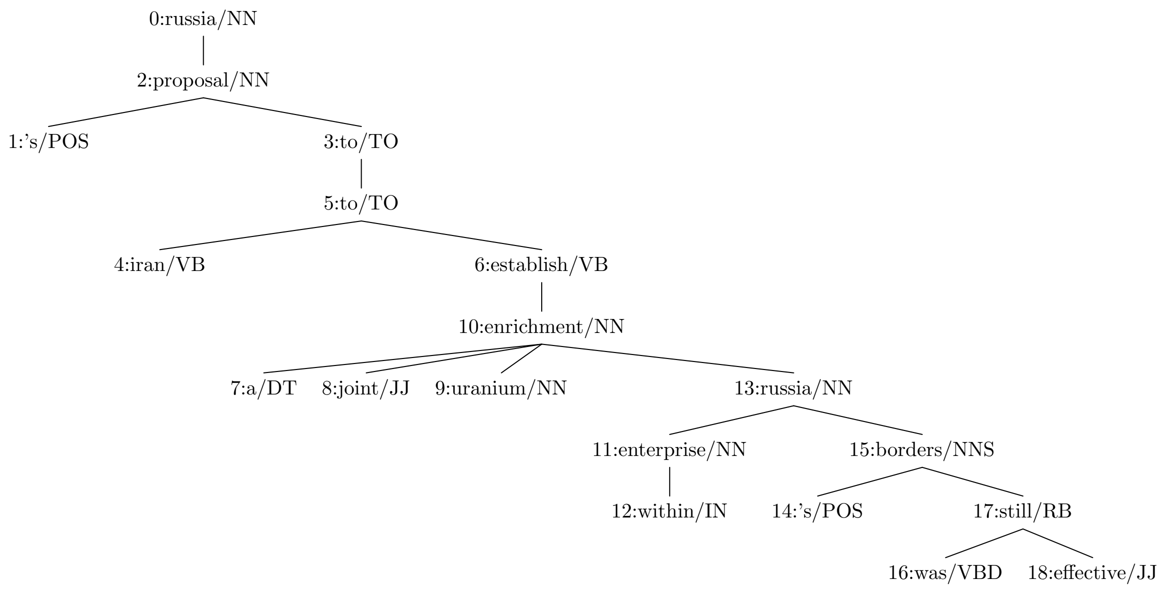


Figure 6-11: Transformed dependency tree with maximum h-gram probability. Transformed from the chain structure.

If we replace λ_h by $\lambda_h P(\mathbf{e})$, then equation 6.10 can be written as

$$P(e|h) = \lambda_h^{-1} c(e|h; \mathbf{e}) \tag{6.11}$$

The value of λ_h^{-1} is not important since it serves only as a reminder that the probabilities must be normalized from the expected counts.

Estimating $c(e|h; \mathbf{e})$ directly by its definition is infeasible given $(l+1)^l$ number of possible parses. However, one can prove that

$$\sum_{\mathbf{h}_\pi(e_1)=0}^{l} \sum_{\mathbf{h}_\pi(e_2)=0}^{l} \cdots \sum_{\mathbf{h}_\pi(e_l)=0}^{l} \prod_{i=1}^{l} P(e_i|\mathbf{h}_\pi(e_i)) = \prod_{i=1}^{l} \sum_{j=0}^{l} P(e_i|h = e_j) \tag{6.12}$$

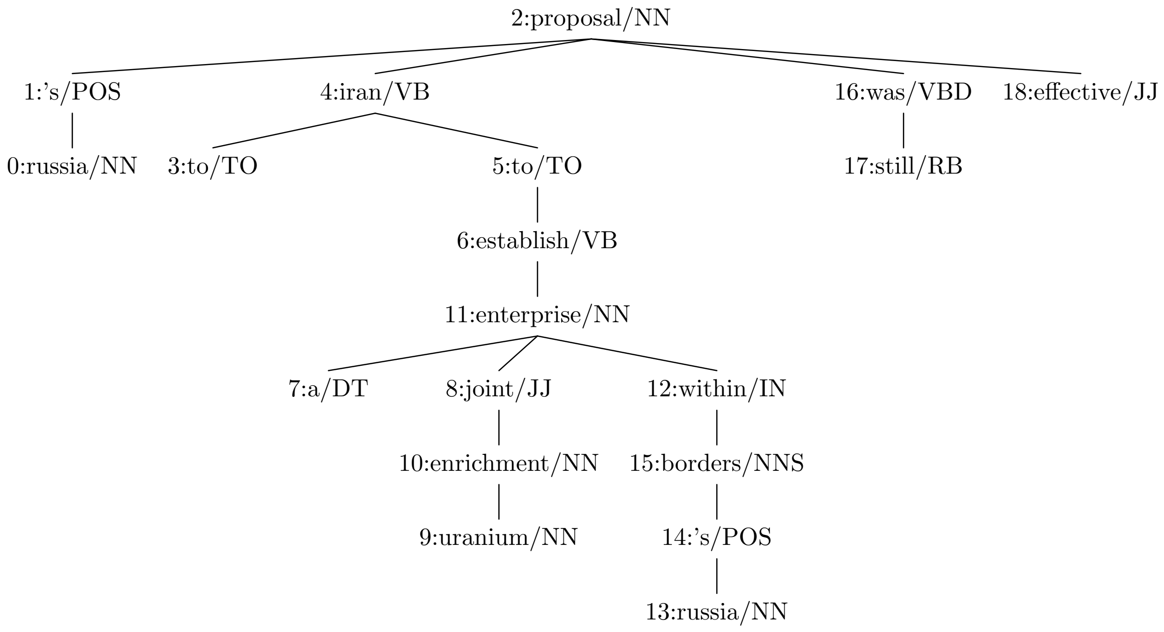


Figure 6-12: Transformed dependency tree with the highest probability. The probability of a word e_i is $\max(P(e_i|\pi_d(i)), P(e_i|\pi_g(i)), P(e_i|\pi_h(i)))$. Transformed from the parser output.

which allows us to rewrite equation 6.2 as:

$$P(\mathbf{e}) = \frac{1}{(l+1)^l} \prod_{i=1}^{l} \sum_{j=0}^{l} P(e_i|h = e_j) \tag{6.13}$$

If we use equation 6.13 to replace equation 6.3 for the Lagrange auxiliary function $\Lambda(P, \lambda)$, set $\frac{\partial \Lambda(P,\lambda)}{\partial P(e|h)=0}$ result in

$$P(e|h) = \lambda_h^{-1} \frac{P(e|h)}{P(e|h=e_0) + P(e|h=e_1) + \cdots + P(e|h=e_l)} \sum_{i=1}^{l} \delta(e, e_i)\delta(h, \mathbf{h}_\pi(e_i)). \tag{6.14}$$

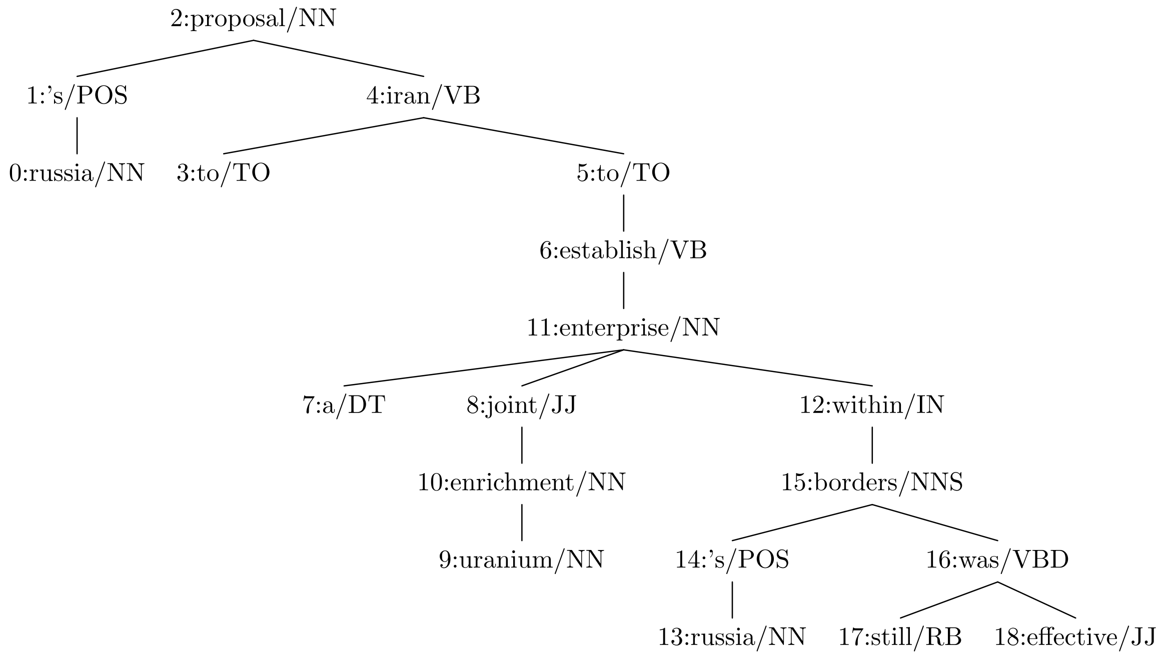


Figure 6-13: Transformed dependency tree with the highest probability. The probability of a word e_i is $\max(P(e_i|\pi_d(i)), P(e_i|\pi_g(i)), P(e_i|\pi_h(i)))$. Transformed from the chain structure.

Comparing this equation with equation 6.10, we find that

$$c(e|h;\mathbf{e}) = \frac{P(e|h)}{P(e|h=e_0)+P(e|h=e_1)+\cdots+P(e|h=e_l)} \sum_{i=1}^{l} \delta(e,e_i)\delta(h,\mathbf{h}_\pi(e_i)). \tag{6.15}$$

The number of operations is now $O(l)$ compared to $O(l^l)$ as suggested by definition 6.9 $P(e|h)$ is re-estimated at the end of each iteration as

$$P(e|h) = \frac{c(e|h)}{\sum_e c(e|h)} \tag{6.16}$$

In implementation, we do not allow a word to be the head word of itself and the expected count for pair $(e|h)$ are summed over all sentences in the training corpus. This is the key difference from the IBM model 1 translation model. Without this constraint, the probability distribution will

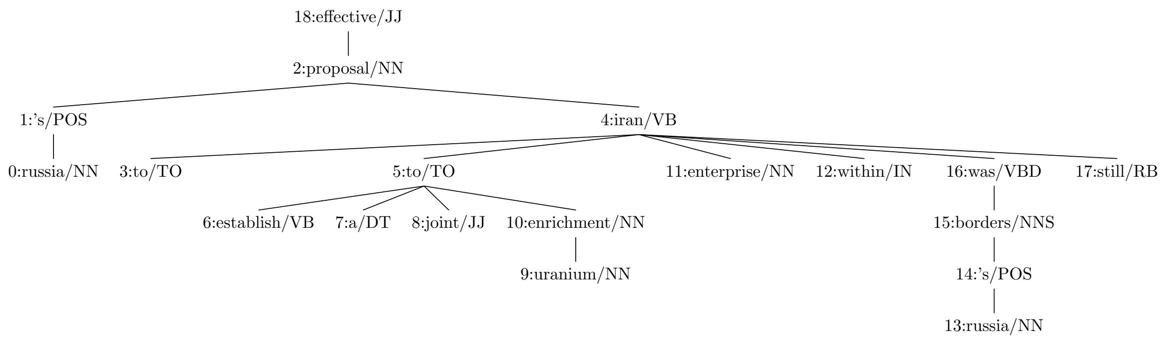


Figure 6-14: Transformed dependency tree with the highest probability. The probability of a word e_i is $\max(P(e_i|e_{i-n+1}^{i-1}), P(e_i|\pi_d(i)), P(e_i|\pi_g(i)), P(e_i|\pi_h(i)))$. Transformed from the parser output.

converge very fast to

$$P(e|h) = \begin{cases} 1 & \text{if } e = h \\ 0 & \text{if } e \neq h \end{cases} \tag{6.17}$$

6.3 Experiments

The structure induction work in this thesis is aimed for language modeling. We evaluate the induced structure and its model by the *gold-in-sands* experiment (Section 7.4). 4 reference translations from the MT03 Arabic test suite are mixed with the 1000-best list generated by the STTK decoder and language model probabilities are calculated for each translation hypotheses and the reference translations. For each testing sentence, we rank the $N + r$ (1004 in this case) translations according to their language model probabilities and report the average rank of the reference translations. As human generated reference translations are gold-standard translation for the testing sentence, a good language model should assign the highest probabilities (lower rank value) to the reference and lower probabilities to MT output. To eliminate the impact of sentence length on language model probabilities, we use sentence level perplexity which is the language model probability averaged on each word in the sentence.

Table 6.2 shows the gold-in-sands experiment results. We compare the reranking results using 5 different language model features: 3-gram language model, 4-gram language model, d3-gram language model trained from the parsed corpus using the MST-parser and $d2$-gram model trained through unsupervised learning. With the unsupervised induced $d2$-gram model, we calculate the probability of a sentence in two different ways:

1. parse each sentence using the cubic-time dependency parser and estimate $P(\mathbf{e})$ as $P(\mathbf{e}|\pi^*$ based on the best parse tree π^*, or
2. estimate the probability of a sentence by summing over all possible structures for this sentence (Eq. 6.13)

Feature ϕ	Model	$P(\mathbf{e})$	Avg. Best Rank	Avg. Rank
3-gram		$\frac{1}{\lvert\mathbf{e}\rvert}\log P(\mathbf{e})$	212.54	471.56
4-gram		$\frac{1}{\lvert\mathbf{e}\rvert}logP(\mathbf{e})$	207.89	467.90
	from MST parsed trees	$\frac{1}{\lvert\mathbf{e}\rvert}logP(\mathbf{e}\lvert\pi^*)$	168.70	432.88
d-gram	unsupervised induction	$\frac{1}{\lvert\mathbf{e}\rvert}logP(\mathbf{e}\lvert\pi^*)$	359.89	602.71
	unsupervised induction	$\frac{1}{\lvert\mathbf{e}\rvert}log\sum_\pi P(\mathbf{e},\pi)$	336.2	585.85

Table 6.2: Comparing the d-gram model from unsupervised structure induction with n-gram language model and d-gram model trained from MST-parsed trees in gold-in-sands experiments.

Not surprisingly, the $d2$-gram model learned through the unsupervised induction performs much worse than the $d3$-gram model derived from MST-parser parsed corpus. However, it is disappointing to see that the unsupervised induced model performs worse than the n-gram models. In the future, we will investigate further in this area to see if a more advanced induction model can improve the induction results.

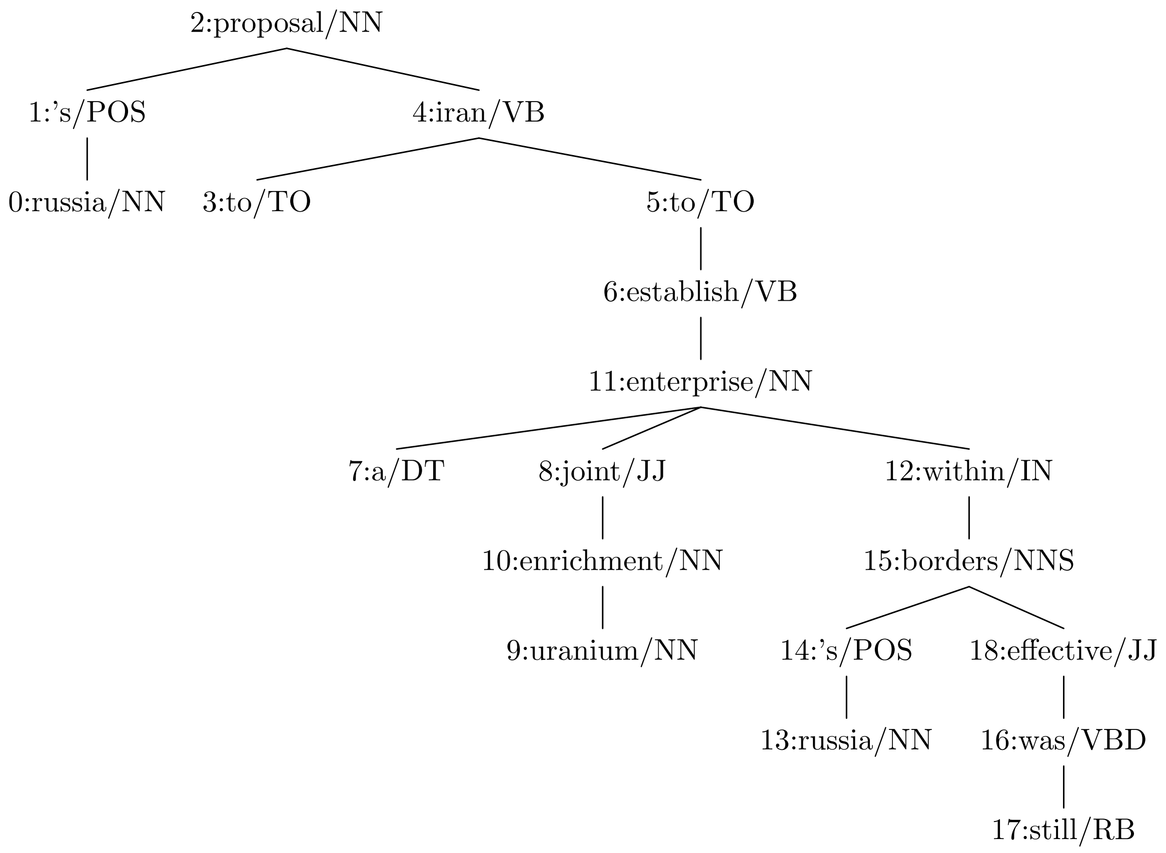


Figure 6-15: Transformed dependency tree with the highest probability. The probability of a word e_i is $\max(P(e_i|e_{i-n+1}^{i-1}), P(e_i|\pi_d(i)), P(e_i|\pi_g(i)), P(e_i|\pi_h(i)))$. Transformed from the chain structure.

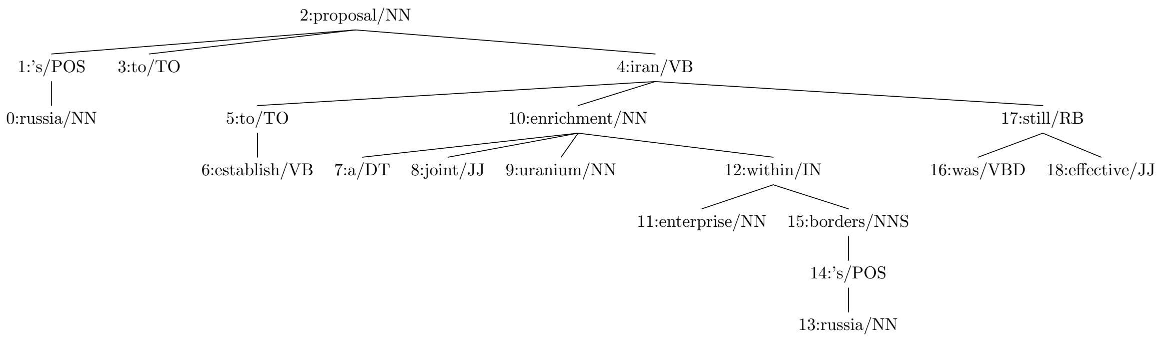


Figure 6-16: Transformed dependency tree with the highest probability. The probability of a word e_i is $\frac{1}{4}(P(e_i|e_{i-n+1}^{i-1})+P(e_i|\pi_d(i))+P(e_i|\pi_g(i))+P(e_i|\pi_h(i)))$. Transformed from the parser output.

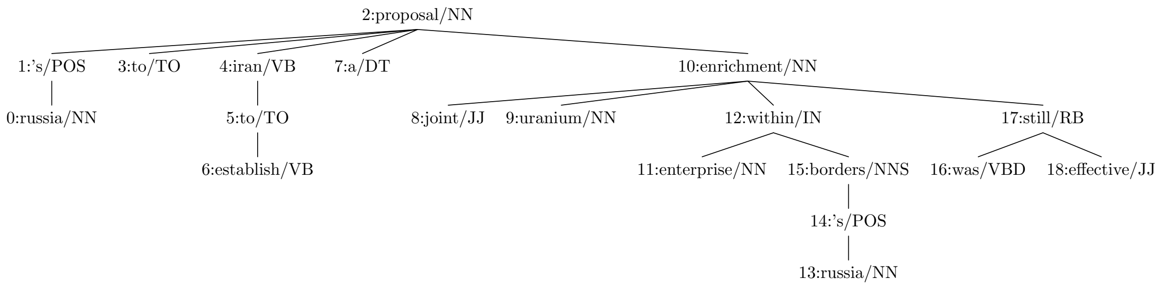


Figure 6-17: Transformed dependency tree with the highest probability. The probability of a word e_i is $\frac{1}{4}(P(e_i|e_{i-n+1}^{i-1})+P(e_i|\pi_d(i))+P(e_i|\pi_g(i))+P(e_i|\pi_h(i)))$. Transformed from the chain structure.

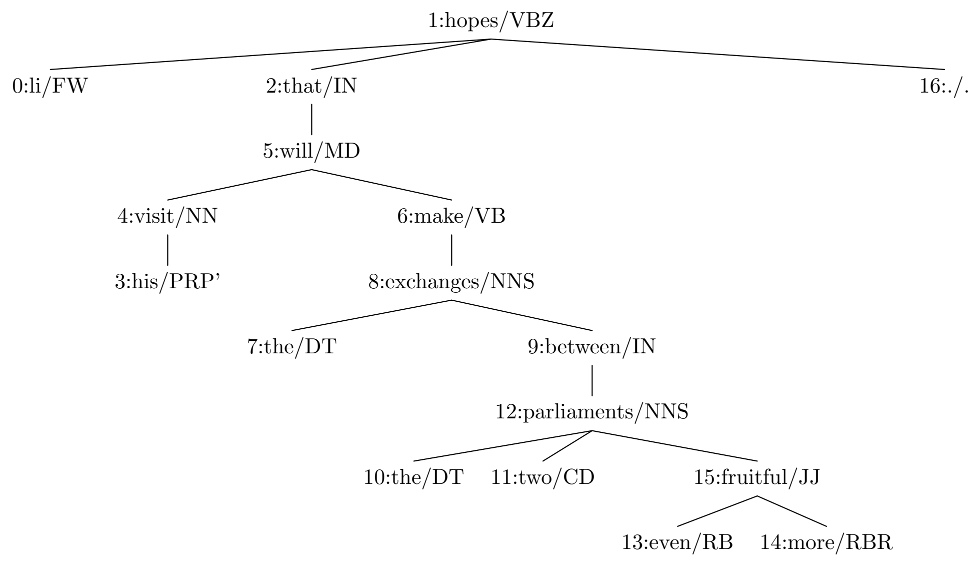


Figure 6-18: Dependency tree by MST parser trained from the treebank.

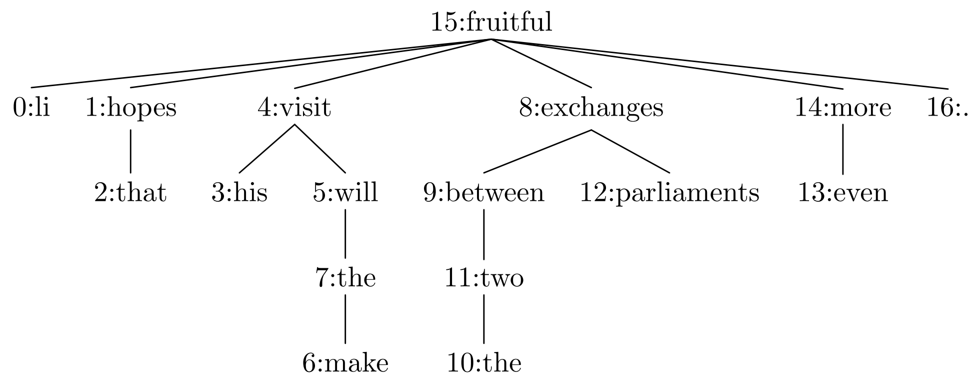


Figure 6-19: Dependency tree by CYK parser with $d2$-gram model from unsupervised training. 3 iterations EM updating.

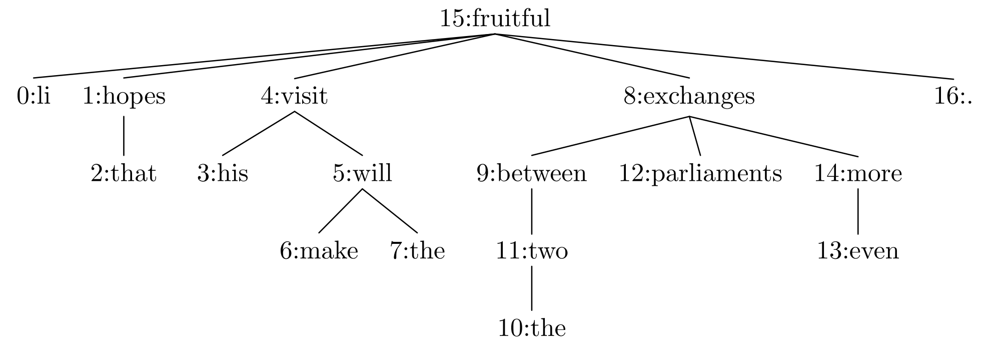


Figure 6-20: Dependency tree by CYK parser with $d2$-gram model from unsupervised training. 6 iterations EM updating.

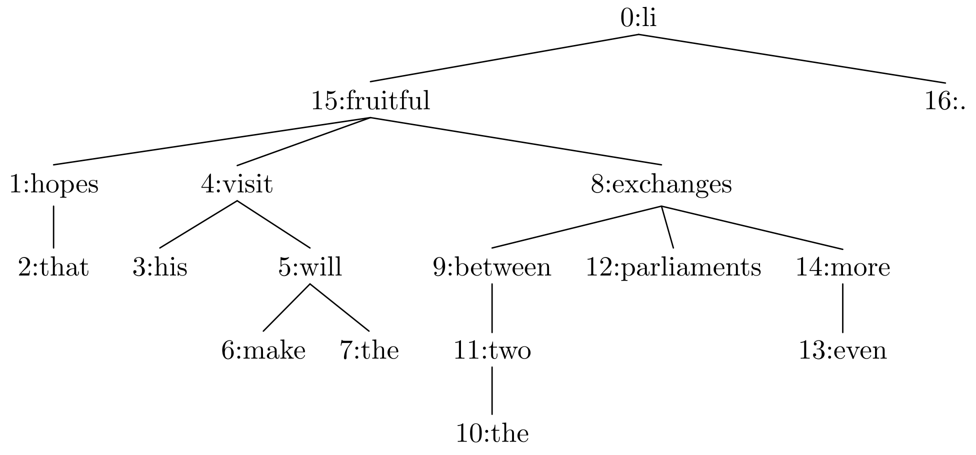


Figure 6-21: Dependency tree by CYK parser with $d2$-gram model from unsupervised training. 9 iterations EM updating.

Chapter 7

Evaluation

Machine translation evaluation itself has been a research topic since the first MT system. The need for MT evaluation is obvious: MT system users need to know how good an MT system is and MT researchers need to know if new development leads to improvement in translation quality. In early days, MT evaluation is mainly subjective, i.e. human judges assign scores to the MT output from various perspectives such as fluency and adequacy. The results of subjective evaluation is usually not repeatable: the same judge could assign a different score to the same MT output next time and of course subjective evaluation is expensive and time consuming.

7.1 n-gram Based Evaluation Metric: BLEU

One revolutionary idea in MT research is the introduction of automatic MT evalution metrics. BLEU [Papineni et al., 2001] is one of the first and still one of the most popular automatic MT evaluation metrics.

The BLEU metric is base on the *modified n-gram precision*, which counts how many n-grams of the candidate translation match with the n-grams of the reference translation(s).

Denote the frequency of an n-gram $\tilde{e}$ in a sentence $\mathbf{e}$ as $C_{\mathbf{e}}(\tilde{e})$, defined as:

$$C_{\mathbf{e}}(\tilde{e}) = \sum_{\tilde{w} \in \mathbf{e}} 1(\tilde{w} == \tilde{e}) \tag{7.1}$$

The *modified n-gram precision* of a translation set as compared to m human reference translation is estimated as:

$$p_n = \frac{\sum_{\text{sent} \in \{Hyp\}} \sum_{n\text{-gram} \in \text{sent}} \min(C_{\text{sent}}(n\text{-gram}), \max_{m=1}^{R} C_{\text{ref}_m}(n\text{-gram}))}{\sum_{\text{sent} \in \{Hyp\}} \sum_{n\text{-gram} \in \text{sent}} C_{\text{sent}}(n\text{-gram})} \tag{7.2}$$

The reason that p_n is called *modified* n-gram precision lies in the fact that the frequency of an n-gram type in the reference is considered when counting the number of matches of the n-gram tokens in the hypothesis. To compute p_n, one first counts the maximum number of times an n-gram occurs in any single reference translation. Next, one clips the total counts of each candidate n-gram by its maximum counts, add these clipped counts up, and divides by the total (unclipped) number of candidate n-grams.

Because BLEU usually uses multiple references, it can not use the concept of "recall" directly. Instead, BLEU uses "brevity penalty" (also know as the "length penalty") to penalize candidate translations that are shorter than their reference translations. For a candidate translation with length c, its brevity penalty (BP)is defined as:

$$BP = \begin{cases} 1, & \text{if} \quad c > r \\ e^{(1-r/c)} & \text{if} \quad c \leq r \end{cases} \tag{7.3}$$

where r is the "closest matching length" among all reference translations.

The final BLEU score is the geometric average of the modified n-gram precision multiplied by the brevity penalty:

$$\text{BLEU} = \text{BP} \cdot \exp(\sum_{n=1}^{N} w_n \log p_n). \tag{7.4}$$

Usually, we use N=4 and $w_n = 1/N$. Several other MT evaluation metrics have derived from BLEU, such as the *NIST MTEval* metric [NIST, 2003] where n-grams are weighted by their information gains, *modified BLEU* [Zhang and Vogel, 2004] where arithmetic average is used to replace the geometric average, and n-gram F-measure [Melamed et al., 2003] where recall is calculated directly.

The key component of BLEU is the averaged n-gram precision. As BLEU is n-gram based, it purportedly favors systems using n-gram based approaches such as statistcal machine translation systems using phrase-based translation model and n-gram language models. MT evaluation campains in recent years have seen syntax-based translation systems such as the Systran MT system and the Sakhr systems with lower BLEU scores than some of the best statistical MT systems even though their human judgement scores are higher. The dominance of BLEU in MT evaluation hinders the development of syntax-driven MT research.

The limitation of BLEU and other n-gram based evaluation metrics are known the machine translation community [Callison-Burch et al., 2006, Zhang et al., 2004]. Figure 7-1 shows a typical example where BLEU fails. In this example, the oracle BLEU-best hypothesis from a 1000-best list for the testing sentence is no better than the baseline translation. It has a very high BLEU score mainly because of a 5-gram *the vote of confidence in* is matched in one of the references.

Ref1	*Pakistani president Musharraf wins vote of confidence in both houses*
Ref2	*Pakistani president Musharraf won the trust vote in senate and lower house*
Ref3	*Pakistani president Musharraf wins votes of confidence in senate and house*
Ref4	*The Pakistani president Musharraf won* **the vote of confidence in** *senate and lower house*
Decoder best H_1	*pakistani president musharraf won the house and the senate confidence vote*
Highest Bleu H^*	*pakistani president musharraf won the senate and speaker of* **the vote of confidence in**

Figure 7-1: Reference translations, best hypothesis (hyp) from the SMT decoder and the hypothesis with the highest BLEU score in the N-best list.

In this thesis, we are moving away from the n-gram based language model and explores the structure in a sentence for better language modeling, it would be consistent to evaluate the translation also in the structured manner.

7.2 Structured BLEU

Inline with the structured language model, especially the x-gram framework, we extend the essential idea of BLEU to include structured information during evaluation.

Assuming that we have the dependency structure for each of the reference translations, besides the immediate context n-gram, S-BLEU also considers the matching of d-grams and g-grams between the hypotheses and the reference translations.

Translation hypotheses are usually not grammatical and the testing sentences are usually not in the same domain as the treebanks which are used to train the parser. Thus, statistical parsers usually fail to find good parses for the translation results. In S-Bleu, we only parse the reference translations by assuming that the parsing quality is reasonably good for human generated translations even though the testing data may be of a different domain than the treebank. Each hypothesis starts with a flat structure, that is, all words depend on the pseudo "root" node in a flat tree. Tree transformations (Section 6.1.1) are applied on this flat structure so that the resulting structure of the hypothesis could be as close as the structure of the human reference translation, or in other words, the hypothesis structure is transformed to get the highest S-Bleu score possible.

Similar to BLEU, S-Bleu metric uses the *modified x-gram precision* which counts how many x-grams of the candidate translation match with the x-grams of the reference translations given their dependency parsing trees. Instead of using the brevity penalty, S-Bleu uses the *average modified x-gram recall* to directly measure the recall of the translation given multiple references.

Denote the modified x-gram precision for the order of n as $p_{x,n}$:

$$p_{x,n} = \frac{\sum_{\text{sent}\in\{Hyp\}} \sum_{x\text{-gram}\in\text{sent}} \min(C_{\text{sent}}(x\text{-gram}), \max_{r=1}^{R} C_{\text{ref}_r}(x\text{-gram}))}{\sum_{\text{sent}\in\{Hyp\}} \sum_{x\text{-gram}\in\text{sent}} C_{\text{sent}}(x\text{-gram})} \tag{7.5}$$

Denote the modified x-gram recall for the order of n as $r_{x,n}$:

$$r_{x,n} = \frac{\sum_{\text{sent}\in\{Hyp\}} \sum_{x\text{-gram}\in\text{sent}} \min(C_{\text{sent}}(x\text{-gram}), C_{\text{ref}^*}(x\text{-gram}))}{\sum_{\text{ref}\in\{\text{ref}^*\}} \sum_{x\text{-gram}\in\text{ref}} C_{\text{ref}}(x\text{-gram})}, \tag{7.6}$$

where ref* is the reference that the translation hypothesis end up with the highest x-gram recall at the sentence level:

S-Bleu score is a linearly weighted sum of all x-gram precision and recall.

$$\text{S-Bleu} = \sum_{x,i} \alpha_{x,i} p_{x,i} + \beta_{x,i} r_{x,i} \tag{7.7}$$

$\alpha_{x,i}$ and $\beta_{x,i}$ are interpolation weights for x-gram of order i. $x = 'n',' d',' g', ...$ and $i = 1, 2, 3, 4, ...$. S-Bleu is thus a family of evaluation metrics. Depending on the value of the interpolation weights, S-Bleu could be essentially n-gram precision, n-gram recall, d-gram F1 score, etc. In this thesis, we use n-gram with n=1,2,3,4; d-gram with d=2,3,4 and g-grams with g=2,3,4 to evaluate translatoins.

7.2.1 Correlation With Human Judgements

The ultimate goal of automatic MT evaluation metrics is to predict how human being like or dislike the MT output. Thus, a good metric needs to correlate well with human judgements to be useful.

Metrics such as BLEU and Meteor [Banerjee and Lavie, 2005] correlates well with human assessments on the set level. One exception is their ranking of the syntax-based systems. Based on the n-gram precision and recall, syntax systems are usually given lower scores by BLEU even though their human assessments could be much higher.

In this section, we calculate the correlation of S-Bleu scores with human judgement at the sentence level. We use the NIST MT03 Arabic-English evaluation set which contains human judgements for MT hypothesis. This set contains 4 human reference translations (system ID: ahd, ahe, ahg and ahi) for each testing sentence. 5 research MT systems (ara, arb, ari, arm and arp) and one COTS MT system (ame) were evaluated.

Judges are native speakers of English. They read one of the reference translations for each sentence and then assign fluency and adequacy scores for the MT output. Judges are instructed to spend, on average, no more than 30 seconds assessing both the fluency and adequacy of a segment. They are also instructed to provide their intuitive assessments of fluency and adequacy and not to delay assessment by pondering their decisions. *Fluency* refers to the degree to which the translation is well-formed according to the grammar of the target language. *Adequacy* refers to the degree to which the translation communicates information present in the original source language text. A good translation should be both fluent and adquate. Both fluency and adequacy are presented in a five-point Likert scale. For fluency measure, the score indicates the translation is: 5 - Flawless English; 4 - Good English; 3 - Non-native English; 2 - Disfluent English; 1 - Incomprehensible.

For the adequacy measure, the score answers the question of "how much of the meaning expressed in the gold-standard translation is also expressed in the target translation?": 5 - All; 4 - Most; 3 - Much; 2 - Little ; 1 - None.

The human judgement is designed in a way that each MT output is evaluted by at least two judges who assign fluency/adequacy scores based on one gold-standard reference translation. To make the comparision fair, we calcuate the automatic MT evaluation scores on the same MT system vs. gold-standard pairs as used by human judges. In other words, the automatic evaluation results are based on single reference translations and are calculated on the sentence level. The scores are more fine-grained than the set-level scores and the correlation with the human scores are much lower than than at the set-level.

Table 7.1 shows the correlation between the automatic evaluation metric and the human judgements at the test set level. The automatic evaluation scores are averaged over the sentence level score over all testing sentences in the test set.

Eval. Metric	Fluency	Adequacy
fluency		0.9675
n-gram prec	**0.9528**	0.9471
n-gram recall	0.9361	0.9848
d-gram prec	**0.9538**	0.9420
d-gram recall	0.9382	**0.9861**
g-gram prec	**0.9543**	0.9496
g-gram recall	0.9286	0.9843

Table 7.1: Corrleation between the automatic MT evaluation score and human judgements at test set level.

Table 7.2 shows the overall correlation between the various automatic evaluation features and the human judgements over 7,956 tuples of <MT System ID, gold reference ID, sentence ID>. The x-gram averaged precision is the arithmetic average of 1,2,3,4-gram precisions, and similarly for the

Eval. Metric	Fluency	Adequacy
fluency		0.6773
n-gram avg. prec	0.2053	0.3072
n-gram avg. recall	0.2166	**0.3309**
d-gram avg. prec	0.2099	0.3011
d-gram avg. recall	**0.2187**	0.3185
g-gram avg. prec	0.2076	0.3034
g-gram avg. recall	0.2108	0.3172

Table 7.2: Correlation between the automatic MT evaluation scores and human judgements at sentence level.

averaged x-gram recall. Table 7.3 shows more fine-grained correlation statistics between automatic evaluation metrics and human judgement *fluency* scores for each individual translation systems. Similar correlations are show in table 7.4 for *adequacy*.

Compared to the correlation at the corpus level, such as the 0.9543 for g-gram precision's correlation with the fluency, the correlation at the sentence level is quite low. BLEU score is based on the n-gram precision. As shown in the table, the low correlation indicates the inability of BLEU to distinguishing good translations at the sentence level.

Though all features do not correlate well with human scores, n-gram recall has the highest correlation with adequacy whereas d-gram recall correlates best with the fluency.

Correlations shown above used all 6 MT systems. We also studied the correlation between the automatic evaluation scores and the human judgements for each individual system.

Corr(Metric, Fluency)	All system	ara	arb	ari	arm	arp	ame(COTS)
n-gram prec	0.2053	0.0346	0.1577	0.1166	0.1007	0.1189	0.1271
n-gram recall	0.2166	0.0323	0.1617	0.1373	0.1291	0.1470	0.1051
d-gram prec	0.2099	**0.0465**	0.1600	0.1337	0.1090	0.1459	**0.1407**
d-gram recall	**0.2187**	0.0409	0.1633	**0.1555**	**0.1345**	**0.1703**	0.1205
g-gram prec	0.2076	0.0344	0.1658	0.1310	0.1089	0.1336	0.1305
g-gram recall	0.2108	0.0312	**0.1692**	0.1363	0.1362	0.1497	0.1112

Table 7.3: Correlation between the automatic MT evaluation scores and human judgement Fluency score at sentence level for each MT systems.

Corr(Metric, Adequacy)	All system	ara	arb	ari	arm	arp	ame(COTS)
n-gram prec	0.3072	0.1577	0.2560	0.2186	0.2079	0.1452	0.2130
n-gram recall	**0.3309**	**0.1836**	0.2541	0.2443	**0.2340**	0.1783	0.1962
d-gram prec	0.3011	0.1593	0.2556	0.2365	0.1984	0.1553	**0.2255**
d-gram recall	0.3185	0.1746	0.2528	**0.2558**	0.2172	**0.1836**	0.2071
g-gram prec	0.3034	0.1432	**0.2653**	0.2354	0.2059	0.1471	0.2092
g-gram recall	0.3172	0.1668	0.2612	0.2401	0.2245	0.1754	0.1911

Table 7.4: Correlation between the automatic MT evaluation scores and human judgement Adequacy score at sentence level for each MT systems.

7.2.2 Hypothesis Dependency Structure Transformation

There are two motivations to transform the dependency tree of a hypothesis translation in the structured Bleu evaluation. First, if we only have parses of the reference translations and want to evaluate the plain-text MT hypotheses in a structured manner, we would bootstrap the structure of the hypothesis as a Markov chain where each word depends only on the word before it and then apply tree-transformation on the chain-structure to maximize the S-Bleu score given the reference tree. Second, when a parser is present such that we could parse the MT hypotheses in the same manner as the references, we could still transform the original parsing trees towards the reference trees for higher S-Bleu scores. In this section we study the impact of tree transformation on S-Bleu scores for these two scenarios.

Transform Parsing Trees From the Original Parser Output

To obtain high accuracy parses for the human reference translation is not an easy task. To obtain high accuracy parses on the SMT output is even harder. The incorrect parse of the MT output could result in wrong evaluation result in S-Bleu. We apply dependency tree transformation (Section 6.1.1) on the original dependency structure of the MT hypothesis to maximize the S-Bleu score compared to the reference trees. Figure 7-2, 7-3 and 7-4 show an example of transforming the original dependency tree of hypothesis "the central america presidents representatives refuse the war against iraq" to maximize the *d*-gram F1 score against the reference tree (Figure 7-2). The F1 score of the original tree (Figure 7-3) is 0.139 whereas the transformed tree (Figure 7-4) is 0.194.

Figure 7-2: Dependency tree of reference translation: “central america vice presidents reject war on iraq”

Figure 7-3: Dependency tree from parser for MT hypothesis: “the central america presidents representatives refuse the war against iraq”: *d*-gram F1 score= 0.139

7.2.3 Transform From Chain Structure

If the parser is not present to parse the MT hypothesis, we can bootstrap the structure of a sentence as a Markov chain where each word depends on its previous word and the first word in the sentence becomes the root of the tree.

Figure 7-5 shows the Markov chain structure of the hypothesis:“the central america presidents representatives refuse the war against iraq.” After applying one transformation of swapping node “presidents” with its parent “america”, the dependency tree becomes Figure 7-6 which now has *d*-gram F1 score of 0.167. Applying another transformation on the tree by swapping the node

Figure 7-4: Transformed dependency tree for MT hypothesis. d-gram F1 score= 0.194

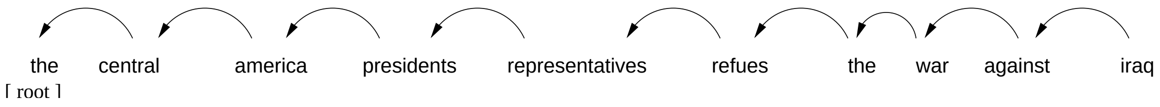


Figure 7-5: Bootstrap the structure of a sentence as a Markov chain.

“presidents” with its new parent “central” turns the tree into Figure 7-7. The transformed tree has d-gram F1 score 0.194 against the reference tree (Figure 7-2). Although the final tree transformed from the chain structure (Figure 7-7) is quite different from the tree transformed from the parser output Figure 7-4, the resulting d-gram F1 scores (0.194) are the same in this case.

Table 7.5 shows the averaged d-gram F1 scores and d-gram recall of hypotheses using the original dependency trees, trees transformed from the parser output and trees transformed from the chain structure. On average, trees transformed from the chain structure have higher scores than the original parser trees. However, trees transformed from the parser output have higher scores than transformed from the chain structure. This is caused by the greedy search in the transformation process where each transformation action must improve the score of the current tree.

Correlation With Human Judgements Using Transformed Trees to Calculate S-Bleu Scores

Table 7.5 shows the correlation between the automatic evaluation scores and the human judgements before and after MT hypothesis dependency trees are transformed to maximize the evaluate scores for each individual hypothesis. As expected, the average d-gram F1 scores increased from 0.2006 to 0.2358 after we transform each dependency tree for higher d-gram F1 scores compared to the

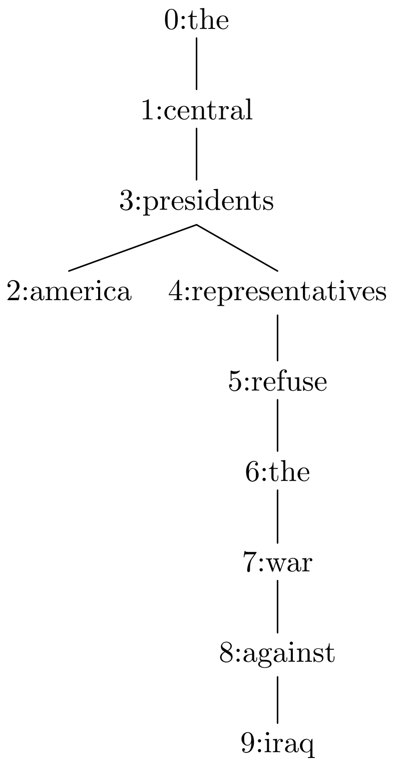


Figure 7-6: After the first transformation. *d*-gram F1 score=0.167.

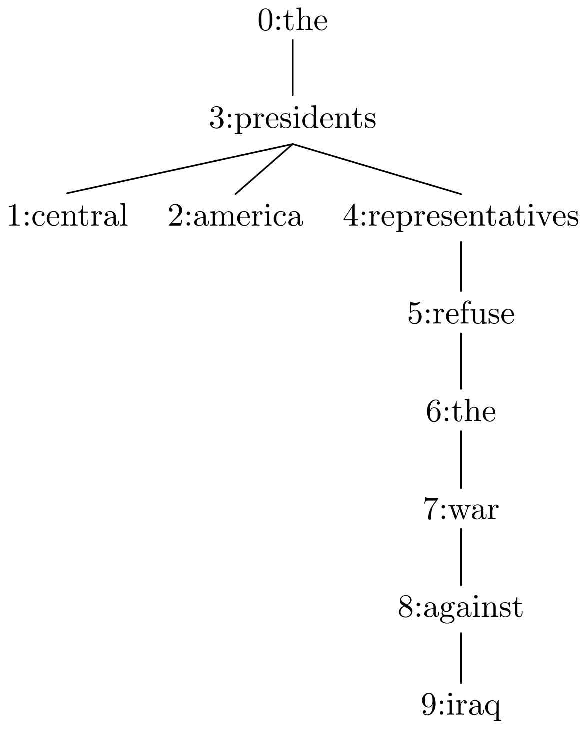


Figure 7-7: After the second transformation. *d*-gram F1 score=0.194.

reference tree. However, the correlation between the *d*-gram F1 score and the fluency score decreased from 0.2189 down to 0.2106. This is also the case for using the *d*-gram recall feature and for the correlation with the adequacy.

7.3 Evaluation of Rerank N-best List Using the Structured Language Model

We apply the structured language model in the N-best list reranking task. The N-best list is generated by the STTK decoder on the Arabic MT03 test set. The top 1000 hypotheses are selected based on their model scores including the translation model, distortion model, n-gram language

(a) Dependency tree of reference translation: "central america vice presidents reject war on iraq"

(b) Dependency tree from parser for MT hypothesis: "the central america presidents representatives refuse the war against iraq": d-gram F1 score= 0.139

model and other models. Although N-best list is a very small subset of the hypotheses spaces and the selection is determined by the non-structured models, it is a reasonably good approximation of the hypotheses spaces Reranking the N-best list is much easier compared to integrate the structured language model into the SMT decoder.

Table 7.6 shows the reranked results of the N-best list. 9 different evaluation metrics are used to evaluate the translation results. To make the comparison fair, original BLEU scores are not used due to its dependency on the length-penalty which is not reliable on the sentence level. However, the n-gram precision score is very similar to the popular BLEU score and as we have shown earlier, correlates with human judgements as well as BLEU scores.

From this table, we can see that n-gram language model performs best in reranking the N-best list when evaluated using n-gram based evaluation metrics (such as BLEU and n-gram F1 score). d-gram model performs best in reranking when translations are evaluated using the d-gram based evaluation metrics.

The scores in Table 7.6 are averaged over 4 reference translations. Evaluation results on different individual reference translations are listed in Appendix B.

Evaluation metric		Avg. score	Correlation with Fluency	Correlation with Adequacy
d-gram F1	dep. trees from parser :	0.2006	0.2189	0.3179
	transform from parser output:	0.2358	0.2106	0.3096
	transform from chain structure:	0.2125	0.2069	0.3058
d-gram recall	dep. trees from parser:	0.2051	0.2188	0.3188
	transform from parser output:	0.2383	0.2111	0.3106
	transform from chain structure	0.2157	0.2086	0.3096

Table 7.5: Correlation between d-gram F1/recall and human judgement fluency/adequacy scores using: original dependency trees from the parser; trees transformed to maximize the evaluation metric starting from the original parse tree, and starting from chain structure and tree t and after hypothesis dependency tree transformation.

7.3.1 Limitations of S-Bleu

Compared to the widely used BLEU and other n-gram based metrics, S-Bleu metrics require the structure of the human references. In this thesis, structure of reference is obtained through a statistical parser. The parser, trained from the WSJ treebanks, is likely to have parsing errors on out-of-domain and/or different genre data. Parsing the reference is time consuming compared to the n-gram precision and recall calculation used in Bleu. To parse the hypotheses using the parser or to transform the chain structure towards the reference trees is also computationally expensive, which makes techniques such as the Minimum-Error-Training (MER) that requires frequent and fast evaluation difficult to be applied in the real system.

7.3.2 Related Work

[Liu and Gildea, 2005] proposed several evaluation metrics using syntactic features to augment BLEU. Five metrics were proposed: the syntax subtree metric: STM and DSTM, the kernel-based subtree metric (TKM and DTKM) and the headword chain based metric.

The "subtree metric" STM calculates the sub syntax tree modified-precision given the phrase structure of reference trees:

$$\text{STM} = \frac{1}{D}\sum_{n=1}^{D}\frac{\sum_{t\in \text{subtrees}_{n}(hyp)}\text{count}_{\text{clip}}(t)}{\sum_{t\in \text{subtrees}_{n}(hyp)}\text{count}(t)} \quad (7.8)$$

	n-gram			d-gram			g-gram		
	prec.	recall	F-1	prec.	recall	F-1	prec.	recall	F-1
Baseline	0.3296	0.3325	0.3284	0.2884	0.2928	0.2883	0.2982	0.3009	0.2967
Oracle-best of:									
BLEU	0.4072	0.3994	0.4005	0.3484	0.3433	0.3436	0.3626	0.3560	0.3565
n-gram F1	**0.4113**	**0.4059**	**0.4060**	0.3515	0.3482	0.3477	0.3641	0.3606	0.3597
d-gram F1	0.3901	0.3850	0.3852	**0.3750**	**0.3711**	**0.3708**	0.3604	0.3559	0.3555
g-gram F1	0.3953	0.3921	0.3912	0.3546	0.3527	0.3516	**0.3850**	**0.3752**	**0.3771**
Reranked by:									
$n3$-gram LM	0.3111	0.3322	0.3184	0.2704	0.2894	0.2771	0.2807	0.3013	0.2876
$n4$-gram LM	**0.3163**	**0.3351**	**0.3225**	0.2728	0.2899	0.2787	**0.2849**	**0.3028**	**0.2906**
$d3$-gram LM	0.3075	0.3242	0.3129	**0.2760**	**0.2921**	**0.2814**	0.2803	0.2942	0.2842
$g3$-gram LM	0.3068	0.3240	0.3123	0.2678	0.2835	0.2730	0.2786	0.2986	0.2850
$max(n3, d3, g3)$	0.3115	0.3233	0.3147	0.2720	0.2837	0.2755	0.2831	0.2956	0.2864

Table 7.6: N-best list reranked by structured LM features and evaluated using structured Bleu features.

where D is the maximum depth of subtrees considered. The DSTM is very similar to STM but it works with the dependency structure of the hypothesis and the reference.

The convolution-kernel-based subtree metric TKM measures the cosine distance between the hypothesis tree and the reference tree over the vector of counts of all subtrees. DTKM is the version for the dependency structure.

The headword-chain-based evaluation (HWCM) metric is very similar to the S-Bleu d-gram precision score. HWCM extracts headword chains from the hypothesis dependency tree and compare them against the headword chains in the reference tree. HWCM is the modified d-gram precision of the hypothesis given the reference tree.

Experiments were run on the 2003 JHU Summer Workshop data and the ACL05 MT evaluation workshop data. Both data sets are for Chinese-English translations. Similar to our finding with S-Bleu, syntax-based metrics correlates better than Bleu with human's fluency judgments.

7.4 Gold-in-sands Experiment

Besides using the n-gram or x-gram based evaluation metrics to calculate the BLEU or S-Bleu scores of the reranked N-best list, we also compare the effectiveness of the structured language model with the n-gram model by *gold-in-sands* experiments. The idea of *gold-in-sands* experiment is to test if a language model (or a feature function in general) assigns higher probabilities to human generated reference translations and lower probabilities to machine translation output. Given the current status of the machine translation technology, the MT output are more worse than the human reference translations even though in some cases MT output may have higher BLEU scores. We assume that when the reference translations are mixed up with the N-best list output, the reference translations should always be the 'best'. The question is, how can we distinguish the good reference translation (gold) from the bad translations (sands). One criteria of a good language model, or in general, a feature function to select the best hypothesis from the N-best list, is the ability to assign higher probabilities/scores to good translations. In the gold-in-sands experiment, we mix r reference translations with the N-best list. Various features are calculated for each of the $N + r$ sentences. The now $N+r$ translations are then ranked according to their feature values. The higher the rank (the lower rank value), the better the sentence is as predicted by the language model.

Table 7.7 shows a gold-in-sand experiment with the 1000-best list generated by the Hiero system on the MT06 Chinese test set broadcasting news portion. There are 4 reference translations for each of the 565 testing sentences. 9 different features are used to rerank the mixed $N + r$ list. We calculate two statistics for each reranked list: the averaged best rank of all r references and the averaged rank of all references. For the t-th testing sentence $\mathbf{f}_t$, denote its four reference translations as $\mathbf{r}_t^{(1)}$, $\mathbf{r}_t^{(2)}$, $\mathbf{r}_t^{(3)}$ and $\mathbf{r}_t^{(4)}$ Denote their ranks in the $N + r$ list based feature ϕ as $R_\phi(\mathbf{r}_t^{(1)})$, $R_\phi(\mathbf{r}_t^{(2)})$, $R_\phi(\mathbf{r}_t^{(3)})$ and $R_\phi(\mathbf{r}_t^{(4)})$. The averaged best rank of r references is:

$$\frac{1}{T}\sum_{t=1}^{T} min(R_\phi(\mathbf{r}_t^{(1)}), R_\phi(\mathbf{r}_t^{(2)}), \ldots, R_\phi(\mathbf{r}_t^{(r)})), \tag{7.9}$$

whereas the averaged rank of all references is:

$$\frac{1}{T}\sum_{t=1}^{T}\frac{1}{r}(R_\phi(\mathbf{r}_t^{(1)}) + R_\phi(\mathbf{r}_t^{(2)}) + \ldots + R_\phi(\mathbf{r}_t^{(r)})). \tag{7.10}$$

Feature ϕ	Avg. Best Rank	Avg. Rank
log(sentLen)	104.88	271.55
ngram	302.11	501.43
avg. ngram	171.78	339.47
dgram	237.74	444.52
avg. dgram	**90.24**	**255.49**
ggram	365.76	557.43
avg. ggram	213.97	400.99
hgram	326.66	538.28
avg. hgram	171.98	360.27

Table 7.7: Gold-in-sands experiment: mixing the reference in the N-best list and rank them using one LM feature.

The decoder parameters are tuned for high BLEU scores. Due to the trade-off between the modified n-gram precision and the brevity penalty, the BLEU-optimized decoder tends to generate shorter translations. Shorter translations have higher n-gram precision than longer ones and the final BLEU scores could still be higher even after being penalized by the brevity penalty. In this particular experiment, the averaged length of the reference translation is 22.78 words whereas the average length of the decoder-best translation is 20.28 words. Thus, as shown in table 7.7, a dummy feature like *log(sentLen)* can rank the reference translations higher (rank 104.88) than most of the MT output. To remove the impact from the sentence length, we also calculated the language model probabilities averaged on words in the gold-in-sands experiments.

Structured language model feature d-gram language model probability outperforms all other features in assigning the highest rank to reference translations. h-gram performs similarly to n-gram model and g-gram model are worse than n-gram.

Table 7.8 shows another gold-in-sands results on the 1000-best list generated by the STTK system [Vogel et al., 2003] for the MT03 Arabic-English test set. There are 4 reference translations for each testing sentences. The top 1000 hypotheses for each testing sentence are used as the N-best list to be mixed with 4 human reference translations in the “gold-in-sands” experiment. The baseline system of using the best hypotheses selected by the decoder gives a BLEU score of 45.76. The oracle-best translation from the 1000-best list gives a much BLEU score of 59.19. 4 reference translations are preprocessed in the same manner as the training data for the language model, including word and punctuation tokenization, lower-casing, and number tagging. We calculate the following features for each of the sentences in the N-best list and each of the 4 references: the structural complexity and the averaged 3-gram, 4-gram, d-gram, g-gram language model probabilities. All language models are trained from the same corpus of 500M words.

The structural complexity of a dependency structure is the total length of the dependency links in the structure. Structural complexities are calculated based on the parsing trees generated by the MST parser [McDonald et al., 2005b]. Structural complexity [Lin, 1996] measures averaged edge length in the dependency tree. Assuming that a grammatical sentence should have a simple structure, words should be close to their dependency parents. Thus a well-formed sentence should have lower structural complexity.

	Decoder Features	3-gram	4-gram	d-gram	g-gram	Structural Complexity
Ref.		212.54	207.89	**168.70**	238.10	269.40
Hyp. with highest BLEU	**489.38**	393.79	**376.14**	422.65	481.24	553.69

Table 7.8: Gold-in-sands experiment: mixing the reference/oracle-bleu-best hypotheses in the N-best list and rank them using one feature.

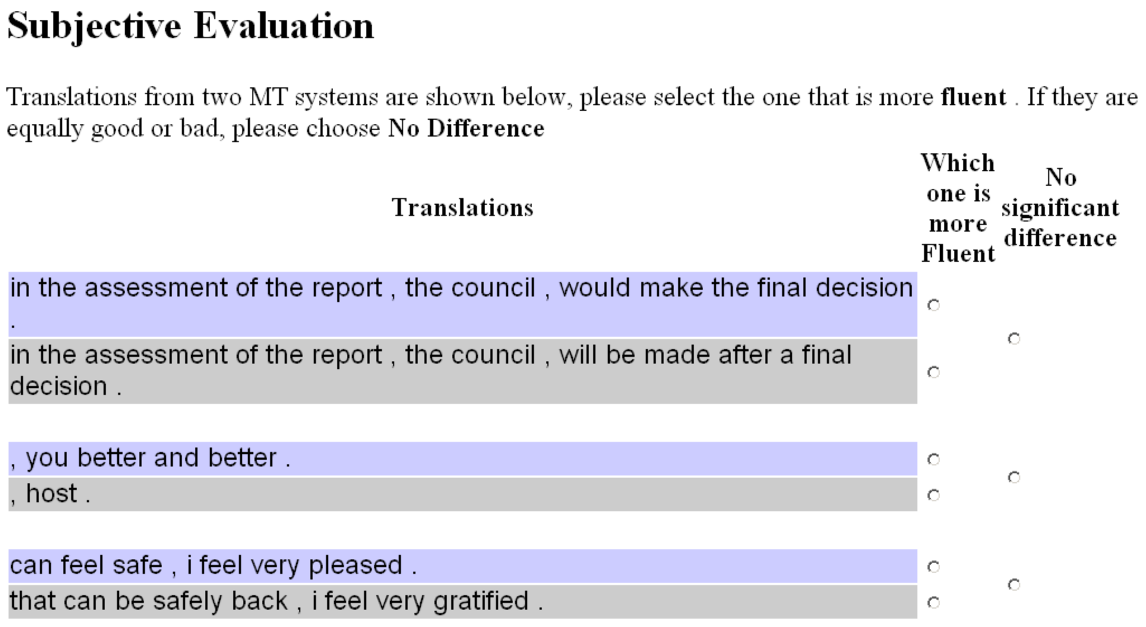
Before running the gold-in-sands experiments, we first calculated the averaged rank of the oracle-BLEU-best hypotheses in the 1000-best list. Surprisingly, the average rank of the oracle-best hypothesis is 489.38, which is in the middle of the 1000-best list. This indicates that the decoder’s internal features do not correlate well with the external evaluation metric such as BLEU. In other words, the decoder does not favor those hypotheses with higher BLEU scores. This is one of the reason why the oracle scores from the N-best list are usually 10+ BLEU points higher than the decoder-best scores.

From the gold-in-sands experiments, we can see that 4-gram language model is better than the

3-gram language model. Reference translations are ranked 207.89 in the $N + r$ list based on their 4-gram language model probabilities compared to average rank of 212.54 by the 3-gram model.

d-gram language model feature ranks the reference translation much higher (168.70) compared to the n-gram model (207.89 by 4-gram). n-gram models perform best in assigning higher ranks to BLEU-best hypothesis, this is inline with our argument and observation that since BLEU metric is essentially n-gram precision score, it is mostly consistent with the n-gram language model.

7.5 Subjective Evaluation



Subjective Evaluation

Translations from two MT systems are shown below, please select the one that is more **fluent** . If they are equally good or bad, please choose **No Difference**

Translations	Which one is more Fluent	No significant difference
in the assessment of the report , the council , would make the final decision .	○	○
in the assessment of the report , the council , will be made after a final decision .	○	
, you better and better .	○	○
, host .	○	
can feel safe , i feel very pleased .	○	○
that can be safely back , i feel very gratified .	○	

Figure 7-8: Screen shot of the subjective evaluation page.

We subjectively evaluated different systems in addition to automatic metrics. Figure 7-8 shows the web-based subjective evaluation page. Translation outputs from two systems A and B are paired on the web page for comparison. The display order of the two systems is randomized for each sentence and the system IDs are hidden from human judges. Human judges are instructed to compare the two translations and choose one that is "significantly more fluent" than the other, or choose "No significant difference" if two translations are equally good or bad.

We randomly select sentences where both system A and B's translations are shorter than 20 words. Long translations, especially from the output of the machine translation systems are difficult for human judges to evaluate since MT output are usually non-grammatical and contain errors almost every where.

System A	A is more fluent	B is more fluent	System B
Decoder-best	24%	28%	BLEU-best
Decoder-best	10%	15%	*n*-gram reranked
n-gram reranked	7%	27%	BLEU-best
h-gram reranked	4%	35%	*n*-gram reranked
d-gram reranked	24%	28%	*n*-gram reranked
g-gram reranked	22%	29%	*n*-gram reranked

Table 7.9: Subjective evaluation of paired-comparison on translation fluency.

Table 7.9 shows the subjective comparison of several system pairs. When comparing the decoder-best (baseline) translations with the BLEU-best (oracle) translation, decoder-best translations are more "fluent" than BLEU-best translations for 24% of testing sentences whereas the latter is more "fluent" than the former for 28% of sentences. These two systems are not distinguishable in fluency for the rest 48% testing sentences. From the table, the n-gram language model performs best among all language models in selecting "fluent" translations from the N-best list. This is different from what we observed in the gold-in-sands experiments where d-gram langauge model performs much better than the n-gram language model in assigning higher probabilities to human reference translations (Table 7.7 and 7.8). Several factors contribute to this result:

- The MST dependency parser is trained on human generated English sentences. The lower parsing accuracy on the MT output influence the structured language model estimations.
- The N-best list is generated using an n-gram based language model. Translation alternatives are not structurally plausible at first place.

- When the translations are bad, human judgements tend to consider sentences with some correct n-grams to be "fluent".

For example, between the two translations for the same testing sentence shown in table 7.10, human judge considers the n-gram reranked translation to be more "fluent" even though both are

n-gram reranked:	remains of victims is expected that the plane will be at around 10:00 pm arrives in hefei .
d-gram reranked:	remains of victims is expected that the plane will be at around 10:00 pm in hefei .

Table 7.10: Comparing the top translations in the N-best list after reranked by n-gram and d-gram language models.

not structurally sound simply because the n-gram *arrives in hefei* makes it locally more fluent. Thus, the subjective evaluation is also biased towards n-gram language model given the current quality of the MT system.

Chapter 8

Future Work

Explore and use structural information in natural language is the future direction for statistical natural language processing. The past 10 years saw great achievements in statistical natural language processing area by using the state-of-the-art machine learning techniques, using more data, and taking advantages of more powerful computers. n-gram based methods play important roles in this era for its simplicity, robustness and effectiveness. As we have shown in this thesis, pushing the limit of n-gram models to large scale with distributed language model and suffix array language model improves the overall system performance. To further improve the statistical natural language processing systems, we have to include the structural information in the model. This thesis proposed the general x-gram framework to model the structure by statistical models. Structured language model is shown to be more discriminative than the n-gram model to distinguish good translations from the bad ones. We see mild improvements when applying the structured language model to rerank the N-best list generated by the n-gram based SMT system. Given the simplicity of the x-gram model which makes it easy to scale up to very large data, there are several interesting area we plan to work on in the future to improve the x-gram model and its applications in statistical NLP.

8.1 Integrating the Structured Language Model Inside the Decoder

d-gram model is significantly better than the n-gram model in the gold-in-sands experiment but only slightly better in reranking the N-best list. In other words, when structurally sound translations are available, d-gram model can better distinguish good translations from bad ones. This indicates that the mild improvement using d-gram models to rerank the N-best list could due to the fact that the N-best list does not contain any structurally sound hypotheses. The decoder used in this thesis is n-gram based, i.e. phrase-based translation model plus n-gram based language models. We are planning to modify the decoder strategy so that the structured language model can be applied in the process of decoding.

8.2 Advanced Unsupervised Structure Induction Models

The unsupervised structure induction model proposed in this thesis, Model 1, is very simple. It does not pose much constraints on the structure and only uses the word surface form. Following the success in word alignment learning in statistical machine translation, we can develop a series of unsupervised structure induction models. Bootstrapping the model using Model 1, one can pose more constraints on the higher level model to estimate more model parameters. In addition, POS and other syntactic and semantic information should be included in the model during the structure induction. Natural language is inherently structured and we believe the structure can be revealed through proper statistical learning.

8.3 Unsupervised Synchronous Bilingual Structure Induction

The unsupervised structure induction induces the dependency structure based on the monolingual (English in this thesis) information only. Given the bilingual corpus, we could apply the same unsupervised structure induction on the foreign language, such as the Chinese and Arabic, to induce the dependency structure. The structural mapping between the the foreign language and English can be learned through the tree alignment models. Such structural translation model addresses the word reordering between the two languages while at the same time maps the foreign words into their English translations.

Empirical results have shown that integrated models work better than aligning structures where the source structure and target structure are obtained using monolingual models. In this sense, we propose to induce a synchronous bilingual parsing model from the data. Such induced bilingual parsing model includes the dependency model for the foreign language $\Pi_{\mathbf{f}}$, dependency model for English $\Pi_{\mathbf{e}}$ and the tree alignment probabilities between the foreign tree and the English tree: A. The model would be trained such that for each sentence pair in the training data $P(\mathbf{f}, \mathbf{e}) = \sum_{\pi_{\mathbf{f}}} P(\mathbf{f}, \pi_{\mathbf{f}}) \sum_{\pi_{\mathbf{e}}} P(\mathbf{e}, \pi_{\mathbf{e}}) a(\pi_{\mathbf{f}}, \pi_{\mathbf{e}})$.

The learned bilingual structure model can then be used in the decoder to translate the source language testing sentence as a tree mapping process.

Appendix A

Notation

$\mathbf{f}$	Foreign sentence to be translated
f_i^j	Substring in $\mathbf{f}$ with words ranging from i to j: $f_i, f_{i+1}, \ldots, f_j$
$\tilde{f}$	A foreign phrase/n-gram
$C(\cdot)$	Count of the entity in parentheses
$\mathbf{e}$	English reference translation; English sentence
$\tilde{e}$	An English phrase (n-gram)
$l(\cdot)$	Length (number of words or characters) of the entity in parentheses
$\mathbf{f_t}$	The t-th testing foreign sentence.
$\mathbf{e_t^{(1)}}$	The "model-best" translation for the t-th sentence.
$\mathbf{e_t^{(r)}}$	The r-th translation in the N-best list of the t-th sentence.
$\mathcal{F}$	Source side of the bilingual corpus, or the source language monolingual corpus
$\mathcal{E}$	Target side of the bilingual corpus, or the target language monolingual corpus
$N(n, r)$	Count of counts: number of n-gram types that have frequency of r.
$\mathcal{E}_d$	The d-th corpus chunk
$\mathbf{d}$	Total number of corpus chunks
$R(\mathcal{E}_d, \mathbf{f}_t)$	In section 3.2.3: Relevance of hypothesis translation $\mathbf{f}_t$ to corpus chunk $\mathcal{E}_d$

$Q(\mathbf{e})$	The nativeness of an English sentence $\mathbf{e}$
V	Vocabulary size
$\cup$	Union of sets
$\cap$	Intersection of sets
$A \setminus B$	Set A minus B
$\|A\|$	Cardinality (number of elements) of a set
$\mathbb{R}$	The set of real numbers
$\mathbb{N}$	The set of natural number
$S_{\tilde{e}}$	The set of sentence IDs where $\tilde{e}$ occurs. $S \subset \mathbb{N}$.

Appendix B

Reranking Results Using Structured Language Model

n-gram prec.	Ref1	Ref2	Ref3	Ref4	Avg. over 4 ref
Baseline	0.3059	0.3883	0.2960	0.3283	0.3296
BLEU-oracle	0.3768	0.5020	0.3493	0.4009	0.4072
n-gram F1 oracle	0.3575	0.5035	0.3656	0.4185	0.4113
d-gram F1 oracle	0.3436	0.4730	0.3475	0.3964	0.3901
g-gram F1 oracle	0.3498	0.4776	0.3509	0.4031	0.3953
Reranked by n3-gram LM	0.2886	0.3652	0.2814	0.3090	0.3111
Reranked by n4-gram LM	0.2933	0.3713	0.2842	0.3162	0.3163
Reranked by d-gram LM	0.2847	0.3654	0.2764	0.3034	0.3075
Reranked by g-gram LM	0.2851	0.3598	0.2783	0.3041	0.3068
Reranked by max(n,d,g)-gram	0.2904	0.3657	0.2824	0.3074	0.3115

Table B.1: N-best list reranked using structured LM and evaluated using n-gram prec.

n-gram recall	Ref1	Ref2	Ref3	Ref4	Avg. over 4 ref
Baseline: decoder features	0.3228	0.3619	0.3062	0.3392	0.3325
BLEU-oracle	0.3862	0.4555	0.3521	0.4037	0.3994
n-gram F1 oracle	0.3687	0.4596	0.3711	0.4244	0.4059
d-gram F1 oracle	0.3541	0.4314	0.3525	0.4022	0.3850
g-gram F1 oracle	0.3622	0.4372	0.3577	0.4113	0.3921
Reranked by n3-gram LM	0.3221	0.3602	0.3078	0.3389	0.3322
Reranked by n4-gram LM	0.3247	0.3637	0.3083	0.3438	0.3351
Reranked by d-gram LM	0.3143	0.3566	0.2983	0.3277	0.3242
Reranked by g-gram LM	0.3148	0.3509	0.3008	0.3296	0.3240
Reranked by max(n,d,g)-gram	0.3144	0.3512	0.3003	0.3271	0.3233

Table B.2: N-best list reranked using structured LM and evaluated using n-gram recall.

n-gram F1	Ref1	Ref2	Ref3	Ref4	Avg. over 4 ref
Baseline: decoder features	0.3122	0.3726	0.2975	0.3311	0.3284
BLEU-oracle	0.3795	0.4757	0.3473	0.3996	0.4005
n-gram F1 oracle	0.3613	0.4788	0.3649	0.4189	0.4060
d-gram F1 oracle	0.3473	0.4496	0.3467	0.3970	0.3852
g-gram F1 oracle	0.3542	0.4549	0.3510	0.4047	0.3912
Reranked by n3-gram LM	0.3024	0.3607	0.2900	0.3204	0.3184
Reranked by n4-gram LM	0.3062	0.3654	0.2919	0.3265	0.3225
Reranked by d-gram LM	0.2968	0.3589	0.2834	0.3125	0.3129
Reranked by g-gram LM	0.2973	0.3531	0.2852	0.3136	0.3123
Reranked by max(n,d,g)-gram	0.3002	0.3564	0.2876	0.3145	0.3147

Table B.3: N-best list reranked using structured LM and evaluated using n-gram F1.

d-gram prec	Ref1	Ref2	Ref3	Ref4	Avg. over 4 ref
Baseline: decoder features	0.2737	0.3364	0.2584	0.2850	0.2884
BLEU-oracle	0.3265	0.4234	0.3022	0.3415	0.3484
n-gram F1 oracle	0.3091	0.4264	0.3135	0.3569	0.3515
d-gram F1 oracle	0.3203	0.4613	0.3349	0.3835	0.3750
g-gram F1 oracle	0.3109	0.4316	0.3162	0.3597	0.3546
Reranked by n3-gram LM	0.2562	0.3157	0.2441	0.2655	0.2704
Reranked by n4-gram LM	0.2578	0.3172	0.2472	0.2691	0.2728
Reranked by d-gram LM	0.2607	0.3215	0.2486	0.2730	0.2760
Reranked by g-gram LM	0.2534	0.3107	0.2445	0.2627	0.2678
Reranked by max(n,d,g)-gram	0.2575	0.3135	0.2500	0.2672	0.2720

Table B.4: *N*-best list reranked using structured LM and evaluated using *d*-gram prec.

d-gram recall	Ref1	Ref2	Ref3	Ref4 Avg. over 4 ref	
Baseline: decoder features	0.2914	0.3162	0.2677	0.2960	0.2928
BLEU-oracle	0.3361	0.3864	0.3060	0.3448	0.3433
n-gram F1 oracle	0.3193	0.3910	0.3195	0.3629	0.3482
d-gram F1 oracle	0.3310	0.4225	0.3413	0.3897	0.3711
g-gram F1 oracle	0.3230	0.3971	0.3236	0.3673	0.3527
Reranked by n3-gram LM	0.2871	0.3128	0.2663	0.2914	0.2894
Reranked by n4-gram LM	0.2871	0.3119	0.2683	0.2925	0.2899
Reranked by d-gram LM	0.2896	0.3145	0.2693	0.2951	0.2921
Reranked by g-gram LM	0.2807	0.3041	0.2648	0.2844	0.2835
Reranked by max(n,d,g)-gram	0.2797	0.3024	0.2679	0.2848	0.2837

Table B.5: *N*-best list reranked using structured LM and evaluated using *d*-gram recall.

d-gram F1	Ref1	Ref2	Ref3	Ref4	Avg. over 4 ref
Baseline: decoder features	0.2805	0.3243	0.2602	0.2882	0.2883
BLEU-oracle	0.3296	0.4025	0.3013	0.3410	0.3436
n-gram F1 oracle	0.3128	0.4065	0.3138	0.3579	0.3477
d-gram F1 oracle	0.3242	0.4395	0.3352	0.3845	0.3708
g-gram F1 oracle	0.3154	0.4123	0.3171	0.3615	0.3516
Reranked by n3-gram LM	0.2690	0.3125	0.2516	0.2755	0.2771
Reranked by n4-gram LM	0.2698	0.3128	0.2542	0.2780	0.2787
Reranked by d-gram LM	0.2726	0.3163	0.2556	0.2814	0.2814
Reranked by g-gram LM	0.2647	0.3055	0.2511	0.2709	0.2730
Reranked by max(n,d,g)-gram	0.2666	0.3063	0.2556	0.2737	0.2755

Table B.6: *N*-best list reranked using structured LM and evaluated using *d*-gram F1.

g-gram prec	Ref1	Ref2	Ref3	Ref4	Avg. over 4 ref
Baseline: decoder features	0.2786	0.3496	0.2702	0.2943	0.2982
BLEU-oracle	0.3384	0.4416	0.3174	0.3530	0.3626
n-gram F1 oracle	0.3206	0.4409	0.3279	0.3671	0.3641
d-gram F1 oracle	0.3152	0.4365	0.3252	0.3645	0.3604
g-gram F1 oracle	0.3339	0.4704	0.3465	0.3893	0.3850
Reranked by n3-gram LM	0.2619	0.3279	0.2575	0.2756	0.2807
Reranked by n4-gram LM	0.2669	0.3327	0.2596	0.2804	0.2849
Reranked by d-gram LM	0.2625	0.3299	0.2542	0.2747	0.2803
Reranked by g-gram LM	0.2610	0.3245	0.2550	0.2740	0.2786
Reranked by max(n,d,g)-gram	0.2667	0.3282	0.2605	0.2770	0.2831

Table B.7: *N*-best list reranked using structured LM and evaluated using *g*-gram prec.

g-gram recall	Ref1	Ref2	Ref3	Ref4	Avg. over 4 ref
Baseline: decoder features	0.2946	0.3255	0.2777	0.3058	0.3009
BLEU-oracle	0.3469	0.4013	0.3187	0.3573	0.3560
n-gram F1 oracle	0.3314	0.4044	0.3320	0.3744	0.3606
d-gram F1 oracle	0.3250	0.3981	0.3292	0.3712	0.3559
g-gram F1 oracle	0.3403	0.4236	0.3455	0.3916	0.3752
Reranked by n3-gram LM	0.2943	0.3251	0.2815	0.3043	0.3013
Reranked by n4-gram LM	0.2964	0.3267	0.2808	0.3073	0.3028
Reranked by d-gram LM	0.2887	0.3193	0.2725	0.2964	0.2942
Reranked by g-gram LM	0.2926	0.3210	0.2781	0.3026	0.2986
Reranked by max(n,d,g)-gram	0.2904	0.3165	0.2778	0.2977	0.2956

Table B.8: *N*-best list reranked using structured LM and evaluated using *g*-gram recall.

g-gram F1	Ref1	Ref2	Ref3	Ref4	Avg. over 4 ref
Baseline: decoder features	0.2843	0.3350	0.2701	0.2974	0.2967
BLEU-oracle	0.3406	0.4186	0.3142	0.3525	0.3565
n-gram F1 oracle	0.3243	0.4200	0.3263	0.3684	0.3597
d-gram F1 oracle	0.3185	0.4146	0.3235	0.3655	0.3555
g-gram F1 oracle	0.3352	0.4436	0.3420	0.3875	0.3771
Reranked by n3-gram LM	0.2750	0.3242	0.2647	0.2864	0.2876
Reranked by n4-gram LM	0.2787	0.3276	0.2657	0.2904	0.2906
Reranked by d-gram LM	0.2728	0.3224	0.2592	0.2823	0.2842
Reranked by g-gram LM	0.2737	0.3202	0.2618	0.2845	0.2850
Reranked by max(n,d,g)-gram	0.2762	0.3202	0.2649	0.2844	0.2864

Table B.9: *N*-best list reranked using structured LM and evaluated using *g*-gram F1.

Bibliography

Rie Kubota Ando and Lillian Lee. Mostly-unsupervised statistical segmentation of Japanese kanji sequences. *Journal of Natural Language Engineering*, 9:127–149, 2003.

J. Baker. Trainable grammars for speech recognition. In J. J. Wolf and D. H. Klatt, editors, *Speech communication papers presented at the 97th Meeting of the Acoustical Society*, pages 547–550, Cambridge, MA, June 1979 1979. Inside-outside algorithm for Stochastic CFG.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W05/W05-0909`.

Srinivas Bangalore. "almost parsing" technique for language modeling. In *Proceedings of the Fourth International Conference on Spoken Language (ICSLP 96)*, volume 2, pages 1173–1176, Philadelphia, PA, USA, Oct 3-6 1996.

Srinivas Bangalore. A lightweight dependency analyzer for partial parsing. *Natural Language Engineering*, 6(2):113–138, 2000. ISSN 1351-3249. doi: http://dx.doi.org/10.1017/S1351324900002345.

Srinivas Bangalore and Aravind K. Joshi. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265, 1999. ISSN 0891-2017.

Jerome R. Bellegarda. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93–108, January 2004. Issue 1: Adaptation Methods for Speech Recognition.

Oliver Bender, Klaus Macherey, Franz Josef Och, and Hermann Ney. Comparison of alignment templates and maximum entropy models for natural language understanding. In *EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, pages 11–18, Morristown, NJ, USA, 2003. Association for Computational Linguistics. ISBN 1-333-56789-0.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, 2007. URL `http://www.aclweb.org/anthology/D/D07/D07-1090`.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2): 263–311, 1993. ISSN 0891-2017.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluation the role of bleu in machine translation research. In *Proceedings of the 11 th Conference of the European Chapter of the Association for Computational Linguistics: EACL 2006*, pages 249–256, Trento, Italy, April 3-7 2006.

Glenn Carroll and Eugene Charniak. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University, Providence, RI, USA, 1992.

Eugene Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 18(4):33–44, 1997.

Eugene Charniak. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France, July 2001.

Eugene Charniak, Kevie Knight, and Kenji Yamada. Syntax-based language models for machine translation. In *Proceedings of the MT Summit IX*, New Orleans, LA, Sep 23-27 2003.

Ciprian Chelba. *Exploiting Syntactic Structure for Natural Language Modeling.* Ph.d., Johns Hopkins University, Baltimore, MD, 2000.

Ciprian Chelba and Frederick Jelinek. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th annual meeting on Association for Computational Linguistics*, pages 225–231, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

Stanley F. Chen. Bayesian grammar induction for language modeling. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 228–235, Morristown, NJ, USA, 1995. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco, 1996. Morgan Kaufmann Publishers.

Zheng Chen, Kai-Fu Lee, and Ming-Jing Li. Discriminative training on language model. In *Proceedings of the Sixth International Conference on Spoken Language Processing (ICSLP 2000)*, volume 1, pages 493–496, Beijing, China, Oct 16-20 2000.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL 2005*, pages 263–270, Ann Arbor, MI, June 2005 2005. ACL.

Noah Chomsky. *Rules and Representations.* Columbia University Press, New York and Chichester, West Sussex, Oct 15 1980. ISBN 0231048270.

Yaacov Choueka. Looking for needles in a haystack, or locating interesting collocational expressions in large textual databases. In *Proceedings of the RIAO*, pages 609–623, Cambridge, MA, March 21-24 1988.

Alexander Clark. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *ConLL '01: Proceedings of the 2001 workshop on Computational Natural Language Learning*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

Michael Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 16–23, Morristown, NJ, USA, 1997. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. Convolution kernels for natural language. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 625–632. MIT Press, 2001. URL `http://dblp.uni-trier.de/rec/bibtex/conf/nips/CollinsD01`.

Michael Collins, Brian Roark, and Murat Saraclar. Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 507–514, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P05/P05-1063`.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

Elliott Franco Drábek and Qiang Zhou. Using co-occurrence statistics as an information source for partial parsing of chinese. In *Proceedings of the second workshop on Chinese language processing*, pages 22–28, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

Jason Eisner. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, October 2000. URL `http://cs.jhu.edu/~jason/papers/#iwptbook00`.

Ahmad Emami, Kishore Papineni, and Jeffrey Sorensen. Large-scale distributed language modeling. In *Proceedings of the 32nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Honolulu, Hawaii, April 15-20 2007.

Robert M. Fano. *Transmission Of Information: A Statistical Theory of Communication*. The MIT Press Classics Series. The MIT Press, Cambridge, MA, March 1961. ISBN 978-0-262-56169-3.

Jianfeng Gao and Hisami Suzuki. Unsupervised learning of dependency structure for language modeling. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 521–528, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3 and 4):237–264, 1953.

Joshua T. Goodman. A bit of progress in language modeling (extended version). Techinical Report MSR-TR-2001-72, Machine Learning and Applied Statistics Group, Microsoft Research, One Microsoft Way, Redmond, WA 98052, August 2001.

Mary P. Harper and Randall A. Helzerman. Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, 9:187–234, 1995.

S. Hasan, O. Bender, and H. Ney. Reranking translation hypotheses using structural properties. In *Proceedings of the EACL'06 Workshop on Learning Structured Information in Natural Language Applications*, pages 41–48, Trento, Italy, April 2006.

Peter A. Heeman. Pos tagging versus classes in language modeling. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 179–187, Montreal, August 1998.

Xuedong Huang, Fileno Alleva, Mei-Yuh Hwang, and Ronald Rosenfeld. An overview of the sphinx-ii speech recognition system. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pages 81–86, Morristown, NJ, USA, 1993. Association for Computational Linguistics. ISBN 1-55860-324-7. doi: http://dx.doi.org/10.3115/1075671.1075690.

Richard A. Hudson. *Word Grammar.* B. Blackwell, Oxford, UK, 1984. URL `citeseer.ist.psu.edu/hudson98word.html`.

R. Iyer and M. Ostendorf. Relevance weighting for combining multi-domain data for n-gram language modeling. *Comptuer Speech and Language*, 13(3):267–282, 1999.

Frankie James. Modified kneser-ney smoothing of n-gram models. Technical report, RIACS, 2000.

F. Jelinek. *Self-organized language modeling for speech recognition*, pages 450–506. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-124-4.

F. Jelinek and R.L. Mercer. *Pattern Recognition in Practice*, chapter Interpolated estimation of markov source parameters from sparse data, pages 381–397. North-Holland, Amsterdam, 1980.

Mark Johnson. Joint and conditional estimation of tagging and parsing models. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 322–329, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163, 1975.

Dan Klein. *The Unsupervised Learning of Natural Language Structure.* PhD thesis, Stanford University, 2005.

Dan Klein and Christopher Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona, Spain, July 2004.

Dan Klein and Christopher D. Manning. Natural language grammar induction using a constituent-context model. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 128–135, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, volume 1*, pages 181–184, 1995.

Bruce Knobe and Gideon Yuval. Some steps towards a better pascal. *Comput. Lang.*, 1(4):277–286, 1976.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, Edomonton, Canada, May 27-June 1 2003.

John Lafferty, Daniel Sleator, and Davy Temperley. Grammatical trigrams: A probabilistic model of link grammar. Technical Report CMU-CS-92-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, Sep. 1992.

Raymond Lau, Ronald Rosenfeld, and Salim Roukos. Trigger-based language models using maximum likelihood estimation of exponential distributions. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Minneapolis, MN, April 1993.

Alon Lavie, David Yarowsky, Kevin Knight, Chris Callison-Burch, Nizar Habash, and Teruko Mitamura. Machine translation working group final report for minds workshops. Technical report, NIST, Nov 06, Feb 07 2006, 2007.

Claudia Leopold. *Parallel and Distributed Computing: A Survey of Models, Paradigms and Approaches.* John Wiley & Sons, Inc., New York, NY, USA, 2001. ISBN 0471358312.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia, July 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P06/P06-1096`.

Dekang Lin. On the structural complexity of natural language sentences. In *Proceedings of the 16th conference on Computational linguistics*, pages 729–733, Morristown, NJ, USA, 1996. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/993268.993295.

Ding Liu and Daniel Gildea. Syntactic features for evaluation of machine translation. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005. URL `http://www.cs.rochester.edu/~gildea/pubs/liu-gildea-eval05.pdf`.

David M. Magerman and Mitchell P. Marcus. Parsing a natural language using mutual information statistics. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, Boston, Massachusetts, July 29-August 3 1990.

Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.*, 22(5):935–948, 1993. ISSN 0097-5397.

M. Manhung Siu; Ostendorf. Variable n-grams and extensions for conversational speech language modeling. *Speech and Audio Processing, IEEE Transactions on*, 8(1):63–75, January 2000. ISSN 1063-6676. doi: 10.1109/89.817454.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, Philadephia, PA, July 6-7 2002.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19(2):313–330, 1993. ISSN 0891-2017.

S. Martin, C. Hamacher, J. Liermann, F. Wessel, and H. Ney. Assessment of smoothing methods and complex stochastic language modeling. In *Proceedings of the 6th European Conference on Speech Communication and Technology*, volume 5, pages 1939–1942, Budapest, Hungary, September 1999.

Ryan McDonald and Joakim Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131, 2007. URL `http://www.aclweb.org/anthology/D/D07/D07-1013`.

Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98, Morristown, NJ, USA, 2005a. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219852.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, Morristown, NJ, USA, 2005b. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1220575.1220641.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. Precision and recall of machine translation. In *Proceedings of NAACL/HLT*, Edmonton, Canada, May 27 - June 1 2003.

Igor A. Mel'čuk. *Dependency Syntax: Theory and Practice.* SUNY Series in Lingusitcs. State University of New York Press, 1988. ISBN 0-88706-450-7.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1964.

Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer, Speech and Language*, 8:1–38, 1994.

NIST. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. Technical report, NIST, http://www.nist.gov/speech/tests/mt/doc/ngram-study.pdf, 2003.

Joakim Nivre. Dependency grammar and dependency parsing. Technical report, Växjö University, School of Mathematics and Systems Engineering, 2005.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, University of Maryland, College Park, MD, June 1999.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. A smorgasbord of features for statistical machine translation. In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-04)*, Boston, 2004. URL `http://www.cs.rochester.edu/~gildea/smorgasbord.pdf`.

Daisuke Okanohara and Jun'ichi Tsujii. A discriminative language model with pseudo-negative samples. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 73–80, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P07/P07-1010`.

Michael A. Olson, Keith Bostic, and Margo Seltzer. Berkeley db. In *ATEC'99: Proceedings of the Annual Technical Conference on 1999 USENIX Annual Technical Conference*, pages 43–43, Berkeley, CA, USA, 1999. USENIX Association.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a method for automatic evaluation of machine translation. Technical Report RC22176(W0109-022), IBM Research Division, Thomas J. Watson Research Center, 2001.

Mark A. Paskin. Grammatical bigrams. In T. Dietterich, S. Becker, and Z. Gharahmani, editors, *Advances in Neural Information Processing Systems 14, Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, pages 91–97, Cambridge, MA, 2001. Cambridge, MA: MIT Press.

Fernando Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 122–127, Morristown, NJ, USA, 1992. Association for Computational Linguistics. ISBN 1-55860-272-0.

Martin Redington, Nick Chater, and Steven Finch. Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22(4):425–469, 1998.

Klaus Ries, Finn Dag Buø, Ye yi Wang, and Alex Waibel. Improved language modeling by unsupervised acquisition of structure. In *Proceedings of ICASSP 95*, volume 1, pages 193–196, Detroit, MI, May 1995.

Brian Roark. Probabilistic top-down parsing and language modeling. *Comput. Linguist.*, 27(2): 249–276, 2001. ISSN 0891-2017. doi: http://dx.doi.org/10.1162/089120101750300526.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 47–54, Barcelona, Spain, July 2004.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, November 1958.

Ronald Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach.* PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, April 1994. TR CMU-CS-94-138.

C. E. Shannon. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 30: 50–64, January 1951.

Daniel Sleator and Davy Temperley. Parsing english with a link grammar. In *Proceedings of the Third International Workshop on Parsing Technologies*, Tilburg, the Netherlands and Durbuy, Belgium, 10 - 13 August 1993.

Noah A. Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P05/P05-1044`.

Zach Solan. *Unsupervised Learning of Natural Languages.* Ph.d., Tel Aviv University, Tel Aviv, ISRAEL, May 2006.

Ray J. Solomonoff. A new method for discovering the grammars of phrase structure languages. In *Proceedings of the International Conference on Information Processing*, Paris, June 15–20 1959.

Ray J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1): 1–22, 1964a.

Ray J. Solomonoff. A formal theory of inductive inference. part ii. *Information and Control*, 7(2): 224–254, 1964b.

Radu Soricut, Kevin Knight, and Daniel Marcu. Using a large monolingual corpus to improve translation accuracy. In *AMTA '02: Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users*, pages 155–164, London, UK, 2002. Springer-Verlag. ISBN 3-540-44282-0.

A. Stolcke and M. Weintraub. Discriminative language modeling. In *Proceedings of the 9th HUB-5 Conversational Speech Recognition Workshop*, Linthicum Heights, MD, 1998.

A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. R. Rao Gadde, M. Plauché, C. Richey, K. Sönmez E. Shriberg, F. Weng, and J. Zheng. The sri march 2000 hub-5 conversational speech transcription system. In *Proceedings of the 2000 Speech Transcription Workshop*, University College Conference Center University of Maryland, May 16-19 2000. NIST.

Andreas Stolcke and Stephen M. Omohundro. Inducing probabilistic grammars by bayesian model merging. In *ICGI '94: Proceedings of the Second International Colloquium on Grammatical Inference and Applications*, pages 106–118, London, UK, 1994. Springer-Verlag. ISBN 3-540-58473-0.

Toshiyuki Takezawa, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world. In *Proceedings of the Third International Conference on Language Resources* *and Evaluation (LREC)*, pages 147–152, Las Palmas de Gran Canaria, Spain, 27 May - 2 June 2002 2002.

Keisuke Tanatsugu. A grammatical inference for context-free languages based on self-embedding. *Bulletin of Informatics and Cybernetics*, 22(3–4):149–163, 1987.

Joseph Turian and I. Dan Melamed. Constituent parsing by classification. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, Vancouver, BC, Canada, October 2005.

Joseph Turian and I. Dan Melamed. Advances in discriminative parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 873–880, Sydney, Australia, July 2006a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P06/P06-1110`.

Joseph Turian and I. Dan Melamed. Computational challenges in parsing by classification. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 17–24, New York City, New York, June 2006b. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W06/W06-3603`.

Joseph Turian, Benjamin Wellington, and I. Dan Melamed. Scalable discriminative learning for natural language parsing and translation. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems (NIPS 2006)*, Vancouver, BC. Canada, December 4-7 2006.

Ashish Venugopal, Stephan Vogel, and Alex Waibel. Effective phrase translation extraction from alignment models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venogupal, Bing Zhao, and Alex Waibel. The CMU statistical translation system. In *Proceedings of MT Summit IX*, New Orleans, LA, September 2003.

Wen Wang and Mary P. Harper. The superarv language model: investigating the effectiveness of tightly integrating multiple knowledge sources. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 238–247, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

Wen Wang, Andreas Stolcke, and Jing Zhang. Reranking machine translation hypotheses with structured and web-based language models. In *Proceedings of 2007 IEEE Automatic Speech Recognition and Understanding Workshop*, Kyoto, Japan, Dec. 9-13 2007.

J. Gerard Wolff. *Categories and Processes in Language Acquisition*, chapter Learning Syntax and Meanings Through Optimization and Distributional Analysis, pages 179–215. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1988.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel copora. *Computational Linguistics*, 23(3):377–403, 1997.

Jun Wu and Sanjeev Khudanpur. Combining nonlocal, syntactic and n-gram dependencies in language modeling. In *Proceedings of Eurospeech'99*, volume 5, pages 2179–2182, Budapest, Hungary, September 6-10 1999.

Peng Xu, Ciprian Chelba, and Frederick Jelinek. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 191–198, Philadelphia, PA, July 7 - 12 2002.

Peng Xu, Ahmad Emami, and Frederick Jelinek. Training connectionist models for the structured language model. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 160–167, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

Mikio Yamamoto and Kenneth W. Church. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Comput. Linguist.*, 27(1):1–30, 2001. ISSN 0891-2017. doi: http://dx.doi.org/10.1162/089120101300346787.

Deniz Yuret. *Discovery of linguistic relations using lexical attraction.* PhD thesis, MIT, Cambridge, MA, 1998. Supervisor-Patrick H. Winston.

Menno Zaanen. Abl: alignment-based learning. In *Proceedings of the 18th conference on Computational linguistics*, pages 961–967, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

Ying Zhang. Suffix array and its applications in empirical natural language processing. Technical Report CMU-LTI-06-010, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Dec. 2006.

Ying Zhang and Stephan Vogel. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proceedings of the Tenth Conference of the European Association for Machine Translation (EAMT-05)*, Budapest, Hungary, May 2005. The European Association for Machine Translation.

Ying Zhang and Stephan Vogel. Measuring confidence intervals for the machine translation evaluation metrics. In *Proceedings of The 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, October 2004.

Ying Zhang, Stephan Vogel, and Alex Waibel. Integrated phrase segmentation and alignment algorithm for statistical machine translation. In *Proceedings of International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE'03)*, Beijing, China, October 2003.

Ying Zhang, Stephan Vogel, and Alex Waibel. Interpreting bleu/nist scores: How much improvement do we need to have a better system? In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, May 2004. The European Language Resources Association (ELRA).

Ying Zhang, Almut Silja Hildebrand, and Stephan Vogel. Distributed language modeling for n-best list re-ranking. In *Proceedings of the 2006 Conference on Empirical Methods in Natural*

Language Processing, pages 216–223, Sydney, Australia, July 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W06/W06-1626`.

Bing Zhao, Matthias Eck, and Stephan Vogel. Language model adaptation for statistical machine translation via structured query models. In *Proceedings of Coling 2004*, pages 411–417, Geneva, Switzerland, Aug 23–Aug 27 2004. COLING.