Unsupervised Structure Induction for Natural Language Processing

Yun Huang

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the School of Computing

NATIONAL UNIVERSITY OF SINGAPORE

2013

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Signature: _____ Date: _____

This thesis is dedicated to my beloved family:

Shihua Huang, Shaoling Ju, and Zhixiang Ren

Acknowledgements

First, I would like to express my sincere gratitude to my supervisors Prof. Chew Lim Tan and Dr. Min Zhang for their guidance and support. With the support from Prof. Tan, I attended the PREMIA short courses on machine learning for data mining and the machine learning summer school, which were excellent opportunities for interaction with top researchers in machine learning. More than being the adviser on my research work, Prof. Tan also provides a lot of help on my life in Singapore. As my co-supervisor, Dr. Zhang made a lot of effort in guiding my research capability from the scratch to being able to carry out research work independently. He also gave me a lot of freedom in my research work so that I can have a chance to develop a broad background according to my interest. I feel so lucky to work with such an experienced and enthusiastic researcher.

During my PhD study and thesis writing, I would thank many research fellows and students in the HLT lab in I^2R for their support. Thank Xiangyu Duan for discussions on Bayesian learning and implementation of CCM. Thank intern student Zhonghua Li for help on implementation of feature-based CCM. Thank Deyi Xiong, Wenliang Chen, and Yue Zhang for discussions on parsing and CCG induction. Thank Jun Lang for his time and efforts for server maintenance. I am also grateful for all the great time that I have spent with my friends in I^2R and NUS.

Finally, I specially dedicated this thesis to my father Shihua Huang, my mother Shaoling Ju, and my wife Zhixiang Ren, for their love and support over these years.

Contents

Abstract

Many Natural Language Processing (NLP) tasks involve some kind of structure analysis, such as word alignment for machine translation, syntactic parsing for coreference resolution, semantic parsing for question answering, etc. Traditional supervised learning methods rely on manually labeled structures for training. Unfortunately, manual annotations are often expensive and time-consuming for large amounts of rich text. It has great value to induce structures automatically from unannotated sentences for NLP research.

In this thesis, I first introduce and analyze the existing methods in structure induction, then present our explorations on three unsupervised structure induction tasks: the transliteration equivalence learning, the constituency grammar induction and the dependency grammar induction.

In transliteration equivalence learning, transliterated bilingual word pairs are given without internal syllable alignments. The task is to automatically infer the mapping between syllables in source and target languages. This dissertation addresses problems of the state-of-the-art grapheme-based joint source-channel model, and proposes Synchronous Adaptor Grammar (SAG), a novel nonparametric Bayesian learning approach for machine transliteration. This model provides a general framework to automatically learn syllable equivalents without heuristics or restrictions.

The constituency grammar induction is useful since annotated treebanks are only available for a few languages. This dissertation focuses on the effective Constituent-Context Model (CCM) and proposes to enrich this model with linguistic features. The

features are defined in log-linear form with local normalization, in which the efficient Expectation-Maximization (EM) algorithm is still applicable. Moreover, we advocate using a separated development set (a.k.a. the validation set) to perform model selection, and measure trained model on an additional test set. Under this framework, we could automatically select suitable model and parameters without setting them manually. Empirical results demonstrate the feature-based model could overcome the data sparsity problem of original CCM and achieve better performance using compact representations.

Dependency grammars could model the word-word dependencies which is suitable for other high-level tasks such as relation extraction and coreference resolution. This dissertation investigates Combinatory Categorial Grammar (CCG), an expressive lexicalized grammar formalism which is able to capture long-range dependencies. We introduce boundary part-of-speech (POS) tags into the baseline model (Bisk and Hockenmaier, 2012b) to capture lexical information. For learning, we propose a Bayesian model to learn CCG grammars, and the full EM and k-best EM algorithms are also implemented and compared. Experiments show the boundary model improves the dependency accuracy for all these three learning algorithms. The proposed Bayesian model outperforms the full EM algorithm, but underperforms the k-best EM learning algorithm.

In summary, this dissertation investigates unsupervised learning methods including Bayesian learning models and feature-based models, and provides some novel ideas of unsupervised structure induction for natural language processing. The automatically induced structures may help on subsequent NLP applications.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In many Natural Language Processing (NLP) tasks, the core process involves some kind of structure analysis. For example, in phrase-based machine translation, the training process would first induce word alignment structures between bilingual sentences. Question answering is another example, in which the knowledge is obtained from the parsed semantic structures. Unfortunately, there are limited resources of annotated structures for NLP. For example, the Penn Treebank (Marcus et al., 1993) has only tens of thousands annotated trees. As a comparison, we can easily obtain billions of sentences from the web. To make things worse, the annotated structures are only available for small number of widely used languages, which limits the NLP researches on other languages. How to induce structures automatically from unannotated sentences has great values.

In this thesis, we investigate and propose new ideas for three structure induction tasks: the transliteration equivalence learning, constituency grammar induction and dependency grammar induction. Evaluation results on annotated test set show effectiveness of our methods.

1.2 Transliteration Equivalence

Proper names are one source of out-of-vocabulary words in many NLP tasks, such as machine translation and cross-lingual information retrieval. They are often translated through transliteration, i.e. translation by preserving how words sound in both languages. For some language pairs with similar alphabets, the transliteration task is relatively easy. However, for languages with different alphabets and sound systems (such as English-Chinese), the task is more challenging.

(a) phoneme representation (b) grapheme representation

Figure 1.1: Transliteration alignments of ⟨smith/史[shi]密[mi]斯[si]⟩. (a) the phoneme representation, in which Chinese characters are converted to Pinyin and English word is represented as phonetic symbols; (b) the grapheme representation, in which literal characters are directly aligned.

Since enumeration of all transliteration pairs is impossible, we have to break word pairs into small transliterated substrings. Syllable equivalents acquisition is a critical phase for all transliteration models. General speaking, there are two kinds of alignments at different representations: phoneme-based and grapheme-based. In the phoneme representations, words are first converted into the phonemic syllables and then the phonemes are aligned. The phoneme systems may be different for source and target languages, e.g. Pinyin for Chinese and phonetic symbols for English. In the grapheme representations, the literal characters in each language are directly aligned. Figure 1.1 illustrates the two representations for aligned transliterated example. Note that the alignments could be one-to-one, one-to-many, many-to-one, and many-to-many. Although many-to-many alignments may be excluded for English-Chinese transliteration, they can be found in other language pairs, e.g. the English-Japanese case (Knight and Graehl, 1998).

Due to the lack of annotated data, inferring the alignments and equivalence mappings for transliteration is often considered as unsupervised learning problems. Simple rule-based models may be used to acquire transliterated equivalences. For instance, for the English-Chinese transliteration task, we may apply rules to find the corresponding character in English word according to the consonants in Chinese Pinyin, and split the English word into substrings. However, rule-based systems often require expert knowledge to specify language-dependent rules, making them hard to handle instances with exceptions or be applied to other language pairs.

Another formalism is the statistical model, which automatically infers alignment structures from given transliterated instances. If there are enough training data, statistical models often perform better than rule-based systems. Furthermore, statistical models could be easily trained for different language pairs. To handle ambiguities, probabilities are assigned to different transliteration alignments in statistical models. The Expectation-Maximization (EM) algorithm is often used to estimate model parameters so as to maximize the data likelihood. One problem of EM is overfitting. In many models (we will see in Section 2.1), if EM is performed without any restriction, the system would memorize all training examples without any meaningful substrings. We propose our Bayesian solution to this problem in Chapter 3.

There are some issues needing to be concerned in transliteration. The first one is that there may be many correct transliteration candidates for the same source word. For example, the name "abare" in English could be transliterated to "阿[a]贝[bei]尔[er]" or "阿[a]巴[ba]尔[er]" in Chinese, and the Chinese transliteration "阿[a]贝[bei]尔[er]" corresponds to "abare" or "abbel" in English. Secondly, name origin may affect the transliteration results. For example, the correct transliterated correspondence of the Japanese-origin name "田[tian]中[zhong]" is "tanaka", where the two words have quite different sounds. In this thesis, we ignore this name origin problem.

1.3 Constituency Grammars

In linguistics, a *constituent* is a word or a group of words that represents some linguistic function as a single unit. For example, in the following English sentences, the noun phrase "**a pair of shoes**" is a constituent acting as a single noun.

She bought **a pair of shoes**.

It was **a pair of shoes** that she bought.

A pair of shoes is what she bought.

There are many kinds of constituents according to their linguistic functions, such as noun phrase (`NP`), verb phrase (`VP`), sentence (`S`), prepositional phrase (`PP`), etc. Usually, the constituents with the same type are syntactically interchangeable. For instance, we may replace the singular noun phrase "**a pair of shoes**" with "**a watch**" without changing the syntactic structure in above examples.

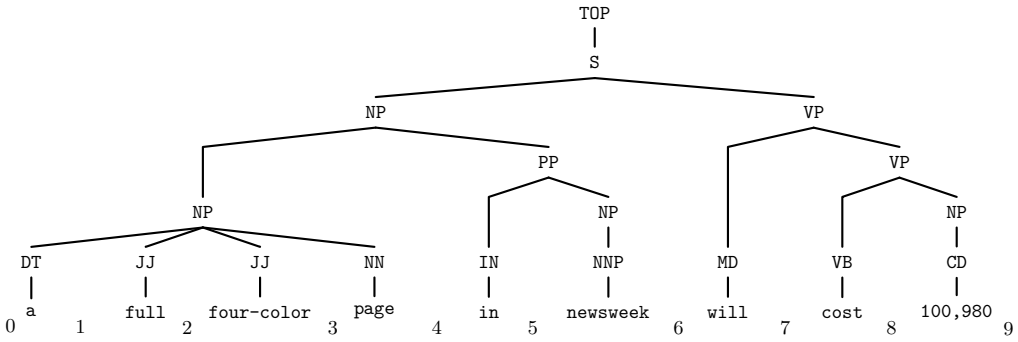


Figure 1.2: A constituency tree example.

The hierarchical structure of constituents forms a constituency tree. Figure 1.2 shows an example, in which the special label `TOP` indicates the root of the tree. Each labeled tree node represents some kind of constituents (`NP`, `VP` ...), and the leaf nodes represent the words. The labels of non-leaf nodes are often called *non-terminals* since they could be expanded in some way, and the words in leaf nodes are *terminals* because the expansion process terminates at these nodes. From this constituency tree, we can extract the

following context-free transformation rules (rules that generate terminals are ignored to save spaces):

$$
\begin{aligned}
\texttt{TOP} &\rightarrow \texttt{S} \\
\texttt{S} &\rightarrow \texttt{NP}\ \texttt{VP} \\
\texttt{NP} &\rightarrow \texttt{NP}\ \texttt{PP} \\
\texttt{NP} &\rightarrow \texttt{DT}\ \texttt{JJ}\ \texttt{JJ}\ \texttt{NN} \\
\texttt{PP} &\rightarrow \texttt{IN}\ \texttt{NP} \\
\texttt{NP} &\rightarrow \texttt{NNP} \\
\texttt{VP} &\rightarrow \texttt{MD}\ \texttt{VP} \\
\texttt{VP} &\rightarrow \texttt{VB}\ \texttt{NP} \\
\texttt{NP} &\rightarrow \texttt{CD}
\end{aligned}
$$

Each rule rewrites (or expands) its left non-terminal (the parent) to the sequence of terminals or non-terminals on the right (the children). The term context-free means that rule applications are independent of contexts and history.

A *constituency grammar* is defined as the tuple of terminals, non-terminals, the special starting symbol, and the set of context-free rewrite rules (Hopcroft et al., 2006). Given constituency grammar, the process of finding grammatical structure from plain string is called *parsing*. Due to the context-free property, dynamic programming algorithms exist for efficient parsing, either from root down to terminals, e.g. the Earley algorithm (Earley, 1983), or in the bottom-up fashion, e.g. the CKY algorithm (Cocke and Schwartz, 1970) for binarized grammars.

To facilitate syntactic analysis, many constituency tree banks have been created in various languages, such as the Penn English Treebank (Marcus et al., 1993), the Penn Chinese treebank (Xue et al., 2005), the German NEGRA corpus (Skut et al., 1998), etc. However, manually creating tree structures is expensive and time-consuming. In this thesis, we are interested in inducing constituency grammars and trees from plain strings. We will review related work in Section 2.2 and propose our model in Chapter 4.

1.4 Dependency Grammars

Constituency grammars perform well for languages with relatively strict word order (e.g. English). However, some free word order languages (e.g. Czech, Turkish) lack a finite verb phrase constituent, making constituency parsing difficult. In contrast, dependency grammars model the word-to-word dependency relations, which is more suitable for languages with free word order.

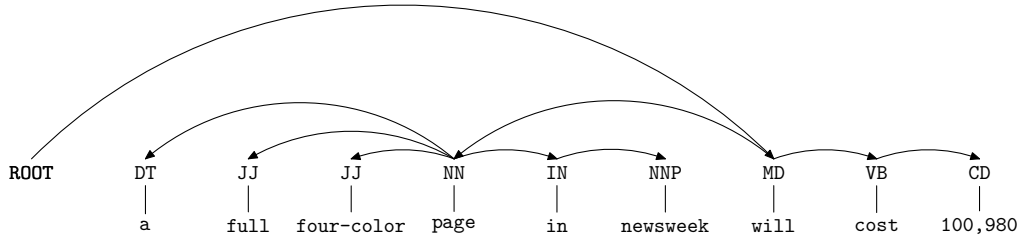


Figure 1.3: A dependency tree example.

In dependency grammar, each word in sentence has exactly one *head word* dominating it in the structure. Figure 1.3 shows a dependency tree in the arc form. Arrows pointing from head to dependents represent dependency relations. The special symbol `ROOT` demonstrates the root of dependency tree that always points to the head word of the sentence (usually the main verb). Arcs may be associated with labels to indicate the relations between the two words, which we omit here for simplicity.

In general, there are two types of relations: the *functor-argument relation* and the *content-modifier relation*. In the functor-argument relation, functor itself is not a completed syntactic category, unless it takes other word(s) as arguments. For example in Figure 1.3, if we remove the word with POS tag "`CD`" from the sentence, the sentence becomes incomplete, since the transitive verb with POS tag "`VB`" must first take an argument as the object. In contrast, if we remove the adjectives with the POS tag "`JJ`" in above example, the sentence remains completed, since the noun "`NN`" could act as a meaningful syntactic category without taking any arguments. In this case, we say that the

adjectives “modify” the noun, which forms the content-modifier relation. We will revisit these concepts in the context of Combinatory Categorial Grammar (CCG) described in Section 1.5. Compared to constituency grammar, lexical information and word order is naturally encoded within dependency grammar.

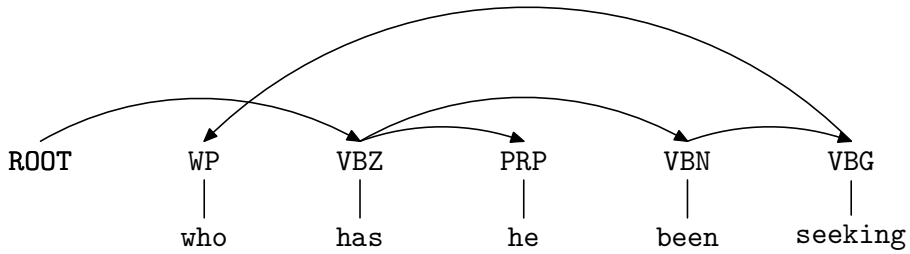


Figure 1.4: A non-projective dependency tree example.

For efficient parsing, many dependency grammars require the dependency trees to be projective, i.e. the arcs can not be crossed. However, this assumption may be violated for languages with free word order. Even for some special structures of English, the projectivity property is not preserved for dependency structure. Figure 1.4 gives example of non-projective dependency structures for the wh-movement structure in English.

Instead of dependency grammar induction, we focus on the induction task of Combinatory Categorial Grammar (CCG) in this thesis. CCG is a more expressive grammar formalism, in which the coordination and the above wh-movement structures are dealt with in an elegant way. We introduce CCG in next section and present models to induce CCG trees in Chapter 5.

1.5 Combinatory Categorial Grammars

Combinatory Categorial Grammar (CCG) is a linguistically expressive lexicalized grammar formalism (Steedman, 2000). Compared to dependency grammars in which words directly act as heads, CCG tree nodes are associated with rich syntactic categories which capture the basic word order and subcategorization. Specifically, the CCG cat-

egories are defined recursively: (1) There are some atomic categories, e.g. S, N; (2) Complex categories either take the form X/Y or X\Y, representing the category that takes category Y as input and outputs the result category X. The forward slash (/) and the backward slash (\) indicate the input category Y follows or precedes the complex category respectively. Note that X and Y themselves may be complex categories too. Parentheses can be used to specify the order of function applications if needed. By default, the slashes are left-associated, e.g. "X\Y/Z" is the shorthand of "(X\Y)/Z". If the order of categories is not important in some cases, we use symbol "|" to represent either the forward slash or the backward slash. The following examples show some common categories in English grammars: N for nouns, NP for noun phrases, S for sentences, (S\NP)/NP for transitive verbs, NP/N for determiners, etc.

The *derivation* of CCG is the sequence of CCG rule applications. There are a few kinds of rule templates defined in CCG. The simplest rules are the forward application (>) and the backward application (<), where the complex category functors take atomic categories as input:

$$\mathtt{X/Y} \quad \mathtt{Y} \quad \Rightarrow \quad \mathtt{X} \quad (\mathtt{>})$$

$$\mathtt{Y} \quad \mathtt{X\backslash Y} \quad \Rightarrow \quad \mathtt{X} \quad (\mathtt{<})$$

The input categories could be complex too, which forms the composition rules:

$$\mathtt{X/Y} \quad \mathtt{Y|Z} \quad \Rightarrow \quad \mathtt{X|Z} \quad (\mathtt{>B}^1)$$

$$\mathtt{Y|Z} \quad \mathtt{X\backslash Y} \quad \Rightarrow \quad \mathtt{X|Z} \quad (\mathtt{<B}^1)$$

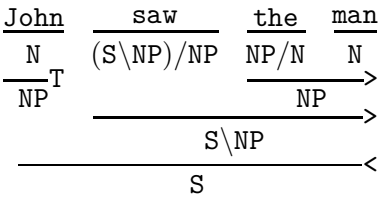
Higher order composition rules can be defined similarly:

$$\mathtt{X/Y} \quad \mathtt{Y|Z}_1|\ldots|\mathtt{Z}_n \quad \Rightarrow \quad \mathtt{X|Z}_1|\ldots|\mathtt{Z}_n \quad (\mathtt{>B}^n)$$

$$\mathtt{Y|Z}_1|\ldots|\mathtt{Z}_n \quad \mathtt{X\backslash Y} \quad \Rightarrow \quad \mathtt{X|Z}_1|\ldots|\mathtt{Z}_n \quad (\mathtt{<B}^n)$$

In a sense, the application rules (> and <) can be regarded as the zero-order case of composition rules ($\mathtt{>B}^0$ and $\mathtt{<B}^0$). Example 1.1 shows the CCG derivations of a declarative sentence. In this example, the lexical category (S\NP)/NP for transitive verb "saw" restricts that the verb must first consume a object noun phrase (NP) on the right to obtain

the intransitive verb category S\NP, then take another noun phrase (NP) on the left as the subject to form sentence. Note that the category N of noun "John" is changed to the category NP using the unary type-changing rule (T). We can see that the CCG lexicons encode rich lexical information as well as the syntactic restriction.



(1.1)

For coordination, CCG assumes that only the same categories can be conjuncted to yield a single category of the same type. In detail, CCG includes a ternary conjunction rule (&). For parsing algorithms (e.g. bottom-up CKY algorithm) that require binary rules, we often use the binarized conjunction rules (>& and <&).

$$\texttt{X} \quad \texttt{conj} \quad \texttt{X} \Rightarrow \texttt{X} \qquad (\texttt{\&})$$

$$\texttt{X} \quad \texttt{X[conj]} \Rightarrow \texttt{X} \qquad (\texttt{>\&})$$

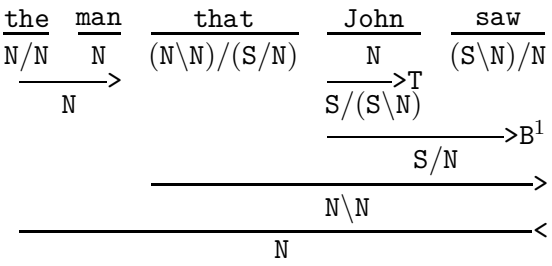
$$\texttt{conj} \quad \texttt{X} \Rightarrow \texttt{X[conj]} \qquad (\texttt{<\&})$$

CCG also includes type-raising rules, which turn arguments into functions over functions-over-such-arguments.

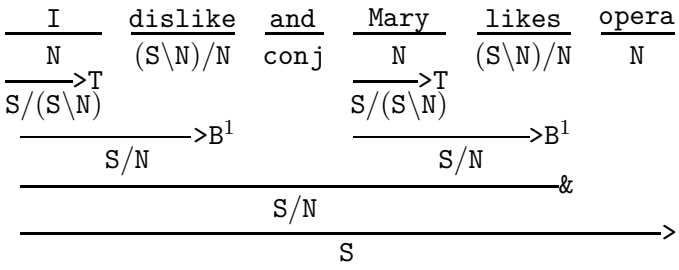
$$\texttt{X} \Rightarrow \texttt{T/(T\backslash X)} \qquad (\texttt{>T})$$

$$\texttt{X} \Rightarrow \texttt{T\backslash(T/X)} \qquad (\texttt{<T})$$

These rules are needed to form some unusual constituents, such as the constituent "John saw" in Example 1.2. In this example, there is no argument on the right to transitive verb "saw" due to the clause structure, so the noun "John" has to be type-raised. Another example of type-raising is the uncommon coordination case (see below), in which two categories of the type S/N are conjuncted.



(1.2)



(1.3)

From example (1.1) and example (1.2), it should be emphasised that the same words have the same lexical categories, although the sentence structures are totally different. This elegant and semantically transparent capture of coordination and extraction of CCG allows recovery of the long-range dependencies and semantics.

Following (Bisk and Hockenmaier, 2012b), we define category X|Y as *functor* if X is different from Y, and category in the form of X|X as *modifier*. In dependency terminology, the functor X|Y corresponds to the head of its argument Y, while the modifier X|X corresponds to the argument of X.

In the formal grammar theory, Combinatory Categorial Grammars are known to be able to generate the language $\{a^n b^n c^n d^n : n \geq 0\}$, and weekly equivalent to Linear Indexed Grammars, Tree-adjoining Grammars, and Head Grammars (Vijay-Shanker and Weir, 1994). As a mildly context-sensitive grammar, CCG models can be efficiently parsed in polynomial time with respect to the sentence length, which makes CCG practical in real tasks. In practice, the "spurious ambiguity" of CCG derivations may lead to an exponential number of derivations for a given constituent. The normal forms of CCG are described in (Eisner, 1996) and (Hockenmaier and Bisk, 2010).

1.6 Structure of the Thesis

The rest of this thesis is structured as follows.

Chapter 2 provides a review of the related unsupervised structure induction approaches, specifically on three induction tasks: transliteration equivalence learning, constituency grammar induction, and dependency grammar induction.

Chapter 3 proposes synchronous adaptor grammar, a general language-independent framework based on nonparametric Bayesian inference, for machine transliteration. The nonparametric priors illustrate the "rich get richer" dynamics, leading to compact transliteration equivalences. The experimental results show that the proposed methods perform better than the EM-based joint source channel model on transliteration tasks for four language pairs.

Chapter 4 presents our explorations on constituency grammar induction. We introduce features to the context-constituent model (CCM), in which various linguistic knowledge could be encoded. Experiments show the proposed model significantly outperforms the CCM, especially on long sentences.

Chapter 5 discusses some improvements on combinatory categorial grammar (CCG) induction. We propose the boundary model and Bayesian learning framework for better CCG induction. The boundary models outperform basic models for full EM, k-best EM and Bayesian inference. Bayesian models achieve better performance than the full EM.

Chapter 6 summarizes contributions of our work and describes some future research directions on these topics.

Chapter 2

Related Work

The rising amount of available rich texts on the web gives an opportunity to improve the performance of many natural language processing tasks. Unfortunately, manual annotations are often expensive and time-consuming. To make things worse, annotated structure corpora are only available for wildly used languages, such as English and Chinese. There are very limited annotated corpora for under-resourced languages. Therefore, it has great value to induce structures automatically from unannotated sentences for NLP research.

Although structure induction remains a challenging problem due to the unsupervised setting, great progress has been made during past twenty years. In this chapter, we first give a quick glance at existing approaches on the transliteration equivalence learning problems, including the monotonic machine translation model and the joint source-channel model. In the second part, we focus on the constituency grammar induction and introduce the constituent-context model, tree-substitution model, and adaptor grammars. Finally, we review the existing approaches on dependency grammar induction, including the dependency model with valence and induction models for combinatory categorial grammars.

2.1 Transliteration Equivalence Learning

Transliteration is defined as phonetic translation across different language pairs (Knight and Graehl, 1998). In the training stage of a transliteration system, finding the alignment between transliterated source and target substrings plays an important role. We give a brief overview of existing models of transliteration equivalence learning in this section.

2.1.1 Transliteration as monotonic translation

Transliteration can be regarded as the monotonic translation problem. Machine transliteration differs from machine translation in two folds: (1) how words sound is preserved during transliteration, while meanings are preserved during translation; (2) there is no reordering problems in transliteration, i.e. the transliterated equivalences are in the same order in both source and target languages. In this view, the word alignment step in Statistical Machine Translation (SMT) (Brown et al., 1993) is adopted to align the transliterated substrings. Similar to SMT, missing sounds are mapped to a special token `NULL`. In SMT, how to derived the internal structure mapping is the key problem of SMT systems. In general, the alignment problem could be categorized by different types of the structures. The simple word-based SMT models using the source and target word pairs as translational equivalences (Brown et al., 1993; Vogel et al., 1996; Moore, 2004; Liu et al., 2009). Advanced word alignment models include: log-linear models (Liu et al., 2005; Moore et al., 2006; Dyer et al., 2011), agreement-based models (Liang et al., 2006; Huang, 2009), Bayesian models (DeNero et al., 2008; Zhao and Gildea, 2010; Mermer and Saraclar, 2011), etc.

Since there is no reordering problem, most of these approaches use simple phrase-based translation models with the word-word alignment. The characters in source and target languages are often aligned using the standard GIZA++ alignment tool[1]. The

[1] http://code.google.com/p/giza-pp/

toolkit runs in source-to-target and target-to-source directions to obtain one-to-many and many-to-one alignments. Then the alignments of two directions are combined with heuristics. Finally, the equivalents are extracted using the standard phrase extraction algorithm (Koehn et al., 2003).

Finch and Sumita (2008) and Rama and Gali (2009) apply the SMT technique for Japanese-English transliteration task. Jia et al. (2009) first use GIZA++ to align characters and then use Moses[2] as decoder to perform transliteration. Another work (Finch and Sumita, 2010b) use a joint multigram model to rescore the output of MT system.

Reddy and Waxmonsky (2009) propose a substring-based transliteration model with Conditional Random Fields (CRFs). In their model, the substrings are first aligned using GIZA++, then the CRF is trained on the aligned substring sequences with the target-side substrings as tags. The similar techniques are also used in (Shishtla et al., 2009). Aramaki and Abekawa (2009) propose to perform monolingual chunking using CRF and then align the bilingual using GIZA++. This model is fast and easy to implement and test, but the performance is not so good.

2.1.2 Joint source-channel models

Li et al. (2004) propose a grapheme-based joint source-channel transliteration model for English-Chinese transliteration, in which the string pairs are generated synchronously. Assuming there are K aligned transliteration units, the probability of string pair $\langle C, E\rangle$ is decomposed as:

$$\begin{aligned} P(\langle C, E\rangle) &= P(\langle c_1, \ldots, c_K, e_1, \ldots, e_K\rangle) \\ &= P(\langle c, e\rangle_1, \ldots, \langle c, e\rangle_K) \\ &= \prod_{k=1}^{K} P(\langle c, e\rangle_k | \langle c, e\rangle_1^{k-1}) \end{aligned} \tag{2.1}$$

[2] http://www.statmt.org/moses/

To reduce the number of free parameters, they assume the transliteration pair only depends on the preceding $n-1$ transliteration pairs. This is similar to the n-gram language model. Then the conditional probability can be approximated

$$P(\langle c,e\rangle_k|\langle c,e\rangle_1^{k-1}) \approx P(\langle c,e\rangle_k|\langle c,e\rangle_{k-n+1}^{k-1}) \tag{2.2}$$

Since the transliteration equivalents are not annotated in training corpus, they perform Expectation-Maximization (EM) learning to infer the substring boundaries. If EM algorithm is performed without restriction, then the model would overfit training data, i.e. each training string pair is memorized without any substring alignments. To overcome this, they restrict that the Chinese side of aligned unit must be one Chinese character. The joint source-channel model shows the state-of-the-art English-Chinese transliteration performance on the standard run of the ACL Named Entities Workshop Shared Task on Transliteration (Li et al., 2009b).

Although the joint source channel models achieve promising results, the overfitting problem of EM needs to be solved carefully. For some language pairs, the one-character restriction is correct in most cases. However, for other language pairs such as Japanese-English, the many-to-many character mappings are common in transliteration equivalents. We show some examples in section 3.4.

To overcome the overfitting problem, Finch and Sumita (2010a) describe a Bayesian model for joint source-channel transliteration model. They formulate the equivalents generating process as the Chinese Restaurant Process (CRP) to learn compact models. (Jansche and Sproat, 2009) and (Nabende, 2009) propose to align syllables based on the weighted finite-state transducer. Zelenko (2009) combine the Minimum Description Length (MDL) training with discriminative modeling for transliteration. Varadarajan and Rao (2009) extend the hidden Markov models and weighted transducers with ϵ-extension for transliteration. We propose the synchronous adaptor grammar, a general nonparametric Bayesian learning framework based on the Pitman-Yor Process (PYP) for transliteration, which we will describe in Chapter 3.

2.1.3 Other transliteration models

System combination often outperforms individual system. Yang et al. (2009) combine the Conditional Random Field (CRF) model and joint source channel model for transliteration. Finch and Sumita (2009) propose to transliterate left-to-right and right-to-left, and finally combine the bi-directional transliterated results. Similar bi-directional transliteration model is also describe in (Freitag and Wang, 2009). Oh et al. (2009) test different strategies to combine the outputs of multiple transliteration engines.

External (monolingual or bilingual) data usually help on the transliteration models. Hong et al. (2009) utilize additional pronouncing dictionary and web-based data to improve the baseline model. Jiang et al. (2009) use manually written rules to convert between grapheme characters and phonetic symbols for transliteration.

Usually, we use the evaluation metrics on the development set to tune model parameters. Pervouchine et al. (2009) propose the alignment entropy, a new evaluation metric without the need for the gold standard reference, to guild the transliteration learning.

Name origin is also an important factor for name transliteration. For example, the written form "田中" is usually transliterated to "`tanaka`" due to its Japanese origin, while it would be transliterated to "`tian zhong`" if treated as a Chinese name. Li et al. (2007) propose a semantic transliteration approach for personal names, in which the name origin and gender are encoded in the probabilistic model. Similarity, Khapra and Bhattacharyya (2009) improve transliteration accuracy using word-origin detection and lexicon lookup.

Usually, the training set of transliterated word pairs are assumed to be available. For some language pairs, however, there are no or small-size available training datasets. (Zhang et al., 2010) and (Zhang et al., 2011) present three pivot strategies for machine transliteration which improve the transliteration results for under-resource language pairs.

2.2 Constituency Grammar Induction

In grammar induction, we want to learn constituency or dependency tree structures from plain strings (words or part-of-speech tags). The induced grammars can be used to construct large treebanks (van Zaanen, 2000), study language acquisition (Jones et al., 2010), improve machine translation (DeNero and Uszkoreit, 2011), and so on. We describe the main approaches on constituency grammar induction in this section.

2.2.1 Distributional Clustering and Constituent-Context Models

From the linguistic point of view, the syntactic categories (such as NP, VP) represent constituents that are syntactically interchangeable. Base on this fact, early induction approaches are based on the distributional clustering. Although clustering methods show good performance on unsupervised part-of-speech induction (Schütze, 1995; Merialdo, 1994; Clark, 2003), distributional similarities do not achieve satisfactory results (Clark, 2001; Klein and Manning, 2001) on unsupervised tree structure induction.

The Constituent-Context Model (CCM) (Klein and Manning, 2002) is the first model achieving better performance than the trivial right-branching baseline in the unsupervised English grammar induction task. Unlike many models that only deal with constituent spans, the CCM defines generative probabilistic models over sequences and contexts for both constituent spans and non-constituent (*distituent*) spans.

In particular, let B be a boolean matrix with entries indicating whether the corresponding span encloses constituent or distituent. Each tree could be represented by one and only one bracketing, but some bracketings are not tree-equivalent, since they may miss the full sentence span or have crossing spans. Define the sequence σ to be the substring enclosed by span, and the context γ to be the pair of preceding and following terminals[3]. The CCM generates sentence S in two steps: first chooses bracketing

[3]For example, in sequence "$_0\texttt{RB}_1\texttt{DT}_2\texttt{NN}_3$", we have $\sigma_{\langle 1,3\rangle} = \langle \texttt{DT NN}\rangle$, and $\gamma_{\langle 1,3\rangle} = \langle \texttt{RB}, \diamond\rangle$. Since

B according to prior distribution $P(B)$, then generates the sentence given the chosen bracketing:

$$P(S, B) = P(B)P(S|B).$$

The prior $P(B)$ uniformly distributes its probability mass over all possible binary trees of the given sentence, and zero for non-tree-equivalent bracketings. The conditional probability $P(S|B)$ is further decomposed to the product of generative probability of sequence σ and context γ for each span $\langle i, j\rangle$:

$$\begin{aligned} P(S|B) &= \prod_{\langle i,j\rangle} P(\sigma_{\langle i,j\rangle}, \gamma_{\langle i,j\rangle}|B_{\langle i,j\rangle}) \\ &= \prod_{\langle i,j\rangle} P(\sigma_{\langle i,j\rangle}|B_{\langle i,j\rangle})P(\gamma_{\langle i,j\rangle}|B_{\langle i,j\rangle}). \end{aligned}$$

From the above decomposition, we can see that given B, the CCM fills each span independently and generates yield and context independently. The Expectation Maximization (EM) algorithm is used to estimate the multinomial parameters $\boldsymbol{\theta}$. In the E-step, a cubic-time dynamic programming algorithm (modified Inside-Outside algorithm (Lari and Young, 1990)) is used to calculate the expected counts for each sequence and context for both constituents and distituents according to the current $\boldsymbol{\theta}$. In the M-Step, the model finds new $\boldsymbol{\theta}'$ to maximize the expected completed likelihood $\sum_B P(B|S, \boldsymbol{\theta}^{old}) \log P(S, B|\boldsymbol{\theta}')$ by normalizing relative frequencies. The detailed derivation can be found in (Klein, 2005).

Although the CCM achieves promising results in short sentences, its performance drops for longer sentences. There are two reasons: (1) CCM models all constituents under only single multinomial distributions, which cannot capture the detailed information of span contents; and (2) long sequences only occur a few times in the training corpus, so the probability estimation highly depends on smoothing. To alleviate these problems,

CCM works on part-of-speech (POS) tags, only POS tags are shown here. The special symbol $\diamond$ represents the sentence boundary.

Smith and Eisner (2004) proposes to generate sequences depending on the length of the spans. Mirroshandel and Ghassem-Sani (2008) describes a parent-based CCM in which the parent spans are also modeled. Golland et al. (2012) applies the local logistic feature based generative model (Berg-Kirkpatrick et al., 2010) to CCM.

In short, distributional clustering and variants of CCM model the distribution of sub-strings. Next, we introduce models that define distributions over sub-trees.

2.2.2 Tree Substitution Grammars and Data-Oriented Parsing

The Tree Substitution Grammars (TSG) are special cases of the Tree Adjoining Grammar (TAG) (Joshi and Schabes, 1997) formalisms without the adjunction operator. The TSG can somewhat be considered as an extension of Context-Free Grammars (CFG) in which the rewriting rules in TSG expand non-terminals to elementary trees rather than symbol strings in CFG. The substitutions happen on the non-terminal leaves in elementary trees. A *derivation* of TSG is a consecutive application of rewriting rules that rewrites (substitutes) the root symbol to terminals. Unlike CFG, the same syntax tree may have more than one derivations in TSG, as illustrated in Figure 2.1. Similar to probabilistic CFG, the probabilistic TSG assigns a probability to each rule in the grammar, and the probability of a derivation is the product of the probabilities of rewriting rules in it. The probability of a syntax tree is the sum of the probabilities of its derivations. Since there exist few annotated TSG corpora, TSG models are usually defined in the unsupervised fashion and derivations are inferred from tree structures, or more challenging from the plain strings.

Data-Oriented Parsing (DOP) is a series of models for tree substitution grammar inference. In the simplest version of DOP (the DOP1 described (Bod, 1998)), tree structures are assumed to be given. Each occurrence of possible subtrees in the treebank is counted as 1. The final probability of a subtree t is computed by normalizing its counts respect to all subtrees with the same parent label. Further researches extend DOP1 to

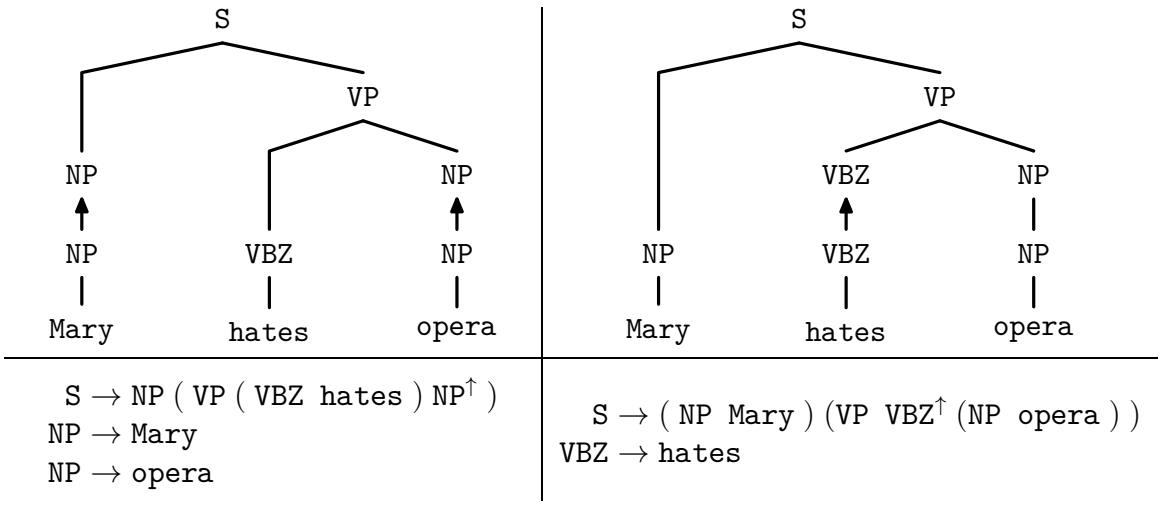


Figure 2.1: Two TSG derivations for the same tree. Arrows indicate the substitution points. The elementary trees used in these two derivations are shown below.

unsupervised parsing and propose the U-DOP model (Bod, 2006b), in which derivations are inferred directly from plain strings rather than tree structures. The key idea of U-DOP is to assign all (unlabeled) binary trees to training sentences and then extract all subtrees from these binary trees. However, the estimation method of DOP1 and other models based on it is biased and inconsistent, which means "the estimated distribution does not in general converge on the true distribution as the size of the training corpus increases" (Johnson, 2002). Following approaches address this problem and propose to use the statistically consistent Maximum Likelihood Estimation (MLE) to learn model parameters (Bod, 2006a; Bod, 2007). Explicitly enumeration of all possible subtrees is intractable, since there are exponential numbers of subtrees given tree structure. Things are even worse if only plain string are given. Most DOP approaches use the method described in (Goodman, 1996; Bod, 2003) to reduce the inference of tree substitution grammar to the inference problem of context-free grammar, in order to avoid the explicit enumeration of subtrees.

The MLE tends to overfit the training data, e.g. each tree is inferred to be generated by single big subtree fragment. Sangati and Zuidema (2011) propose the double-DOP in

which only subtrees occur at least twice in training corpus are modeled. This criterion excludes a large amount of "big" subtree fragments which reduces computation cost and alleviates the overfitting problem as well. Bayesian models for TSG provide systemic solutions to the overfitting problem of MLE (Post and Gildea, 2009; Cohn et al., 2009; Cohn and Blunsom, 2010; Cohn et al., 2010). In Bayesian models, sparse priors (usually the nonparametric Pitman-Yor Process (PYP) priors) are integrated into the model to enforce simple models and encourage common linguistic constructions. Inferences are usually based on sampling, in which only a small fraction of subtrees are stored in cache which avoids the exponential enumeration problem. These models achieve the state-of-the-art grammar induction results.

Tree substitution grammars encode rich information about the tree structures. Compared to CCM with constituents modeled, TSG is more expressive that both contiguous and non-contiguous phrases are modeled. However, one shortcoming of TSG models is the high model complexity with high computation cost, as well as the implementation difficulty for such models.

2.2.3 Adaptor grammars

Adaptor Grammars (AGs) provide a general framework for defining nonparametric Bayesian models based on probabilistic CFGs (Johnson et al., 2007b). In adaptor grammars, additional stochastic processes (named adaptors) are introduced to allow the expansion of an adapted symbol to depend on the expansion history.

In practice, adaptor grammars based on the Pitman-Yor process (PYP) (Pitman and Yor, 1997) are often used in inference. The nonparametric priors let the expansion of nonterminals depend on the number of subtrees stored in cache during sampling. With suitable choose of parameters, the PYP demonstrates a kind of "rich get richer" dynamics, i.e. previous sampled values would be more likely sampled again in following sampling procedures. This dynamic is suitable for many machine learning tasks since they

prefer sparse solutions to avoid the over-fitting problem. Since many existing models could be viewed as special kinds of probabilistic CFG, adaptor grammars give general Bayesian extension to them.

One limitation of adaptor grammars is that the nonterminals in adaptor grammars cannot be recursively defined (i.e. `NP` cannot be expended to another `NP` in one or more induction steps), which restricts the usability of adaptor grammars for inducing natural recursive tree structures. Even so, adaptor grammars have been widely used in various NLP tasks such as topic modeling (Johnson, 2010), perspective modeling (Hardisty et al., 2010), morphology analysis and word segmentation (Johnson, 2008; Johnson and Goldwater, 2009; Johnson and Demuth, 2010), and native language identification (Wong et al., 2012). We will revisit the adaptor grammar and propose extensions in Chapter 3.

2.2.4 Other Models

Seginer (2007) describes a novel structure named the Common Cover Links (CCL) and an unsupervised incremental learning algorithm to induce constituency trees from plain text[4]. Compared to dependency structure, the CCL parser is incremental and extremely fast for both learning and parsing. However, CCL is a model based on heuristics instead of probabilistic algorithm, which makes it hard to extend.

Ponvert et al. (2011) focus on the simpler unsupervised chunking task and proposes a cascaded finite-state model[5]. They use Hidden Markov Model (HMM) and a generalization named Probabilistic Right Linear Grammar (PRLG) (Smith and Johnson, 2007) to label words with $\{\texttt{B}, \texttt{I}, \texttt{O}, \texttt{S}\}$ tags (standing for *Beginning* word, *Inside* word, *Outside* word, and *Single* word of chunks). After determining the phrase boundaries, they choose the most frequent word in each phrase to represent that chunk, repeat the induction steps, and finally obtain the hierarchical structures.

[4] http://www.seggu.net/ccl/
[5] http://elias.ponvert.net/upparse

2.3 Dependency Grammar Induction

Lexical information is useful for supervised constituency parser (Collins, 1997; Charniak, 2000), which may also show benefits in unsupervised grammar induction. The lexical dependencies, such as the head-argument function, modification relation and coordination structures, are directly modeled in dependency grammars. In this section, we review the popular unsupervised dependency grammar induction models.

2.3.1 Dependency Model with Valence

Klein and Manning (2004) propose a simple head-outward dependency model, named Dependency Model with Valence (DMV), where the valence is modeled using a special `STOP` token. The generative process begins at the `ROOT` of the dependency tree. Each head generates its dependents on left side and right side independently. On each side, words are generated in sequence, and finally a `STOP` is generated. The above generative step repeats until the whole sentence is covered. Specifically, when generating a word, the decision whether to terminate (generate `STOP`) is made according to $P_{\texttt{STOP}}(\texttt{STOP}|h, dir, adj)$, where h is the head word, dir is the direction (left or right), and adj is a binary variable indicating whether or not an argument on current side has already been generated. If we decide to generate `STOP`, then no more symbols are generated on that side. Otherwise, the dependent a is chosen according to $P_{\texttt{CHOOSE}}(a|h, dir)$, which is independent of the variable adj. Expectation-Maximization (EM) algorithm is used to estimate the model parameters. With a smart initialization (the ad-hoc "harmonic" completion), the DMV outperforms the trivial right-branching baseline (Klein and Manning, 2004).

Following models based on DMV mainly improve the estimation procedures. Smith and Eisner (2005) propose the contrastive estimation for DMV, in which the probabilities of observed sentences are estimated, conditioned on heuristically constructed neighborhoods (as implicit negative evidences). They also describe two annealing techniques for

better initialization (Smith and Eisner, 2006). Headden III et al. (2009) introduce lexical information into DMV and show how it can be leveraged via smoothing. Cohen and Smith (2009) propose a Bayesian model for DMV with shared logistic normal prior distributions. Spitkovsky et al. (2010a) and Spitkovsky et al. (2010b) compare the performance of the traditional full EM and the Viterbi EM, and find that the Viterbi EM with good smoothing values can achieve better performance than full EM. Naseem et al. (2010) uses manually-specified linguistic-motivated rules in dependency grammar induction. Variational Bayesian method is used to estimate the parameters.

The constituency grammars and dependency grammars capture different aspects of language. Klein and Manning (2004) propose a combined model of CCM and DMV, which outperforms each component in most experiment settings. Blunsom and Cohn (2010) and Cohn et al. (2010) describe methods to represent dependency grammar in the tree substitution grammar formalism and use Bayesian TSG induction to perform dependency grammar induction.

2.3.2 Combinatory Categorial Grammars

As described in Section 1.5, the Combinatory Categorial Grammar (CCG) encodes dependency relations and functor arity into the syntactic categories. CCG formalisms provide a more syntax-meaningful representation, especially for long-range dependencies (Steedman, 2000).

The first type of CCG induction system is the fully unsupervised models, in which no linguistic heuristics are assumed. Osborne and Briscoe (1997) propose an unsupervised learning model for CCG induction. They consider part-of-speech tags as atomic categories, and construct more complex categories using slashes. The first step of their method is to create a labeled binary tree for each part-of-speech tag sequences in a greedy, bottom-up, incremental manner. The label of each inner node is the label of either the left or right sub-node. To avoid overfitting, they apply the Minimum Description

Length (MDL) principle to learn compact grammars with minimal length of hypothesis and minimal length of data encoded in the hypothesis. In the second step, the categories and grammar rules are read off the built trees with frequency counts for further usage. They do not consider the coordination structure or punctuations in their model. Experimental results of their proposed model are not as great as they might be, although they outperform the EM baseline. Ponvert (2007) presents a genetic algorithm to learn CCG categories and grammars. However, their experiments do not show promising results.

Since it is difficult to infer syntactic categories from plain strings, many researches introduce manually written rules to guide the induction procedures. Watkinson and Manandhar (1999) describe an unsupervised approach to learn CCG lexicons. At beginning, the learner is provided with a set of manually defined CCG lexicons. In each step, the parser with current lexicon and rules is used to parse training sentences. Then k-best parses are selected and used to modify lexicons. Experiments on small datasets show the effectiveness of their method. Boonkwan and Steedman (2011) create a framework to describe language characteristics using 30 questions, such as the order of subject, verb, direct object and indirect object, etc. For each language, they encode the answers to those questions into CCG categories and use them to prune search spaces. Their methods achieve the state-of-the-art results on different languages.

Another successful approach that achieves good results without specifying too much linguistic knowledge is the model proposed in (Bisk and Hockenmaier, 2012b). In their grammar, there are only two atomic categories allowed, `N` (nouns or noun phrases) and `S` (sentences), together with a special conjunction category `conj`. The first stage is the lexicon and grammar generation stage. They specify atomic categories to part-of-speech tags initially, and use an iterative algorithm to create more complex lexical categories. Then, the training sentences are parsed using the created lexicons and CCG rules. The basic probabilistic model described in (Hockenmaier and Steedman, 2002) is used in their experiments. They compare various EM settings (full EM, Viterbi EM, and k-best

EM) and find that the k-best EM could achieve best performance. They report the state-of-the-art results for unsupervised dependency grammar induction. In Section 5.1, we give a detailed description of their lexicon and grammar generation method. We propose to use boundary words and Bayesian learning to improve their models, which will be presented in Chapter 5.

2.4 Summary

In this chapter, we have reviewed some existing approaches on three unsupervised structure induction tasks: the transliteration equivalence learning, the constituency grammar induction and the dependency grammar induction. For the transliteration task, we explore the joint source channel model and propose the nonparametric Bayesian extension based on synchronous adaptor grammars in Chapter 3. For the constituency grammar induction, we focus on the simple CCM and present the feature-based CCM in Chapter 4. For the dependency grammar induction, we propose to use boundary word and Bayesian learning for the CCG induction in Chapter 5.

Chapter 3

Synchronous Adaptor Grammars for Transliteration

We focus on the joint source-channel model (Li et al., 2004) for transliteration in this chapter, since it is one of the state-of-the-art models for English-Chinese transliteration (Li et al., 2009a).

As mentioned in previous chapters, this model aims to maximize the likelihood of training data by the Expectation-Maximization (EM) algorithm. However, the EM algorithm may overfit the training data by memorizing the whole training instances. As a result, only single Chinese character is allowed in the syllable mappings in their English-Chinese transliteration experiments. However, the single-character restriction is not always true for other language pairs.

In this chapter, we propose Synchronous Adaptor Grammar (SAG), a novel nonparametric Bayesian learning approach based on the Pitman-Yor process (Pitman and Yor, 1997), for machine transliteration. This model provides a general framework to automatically learn syllable equivalents without heuristics or restrictions. The proposed model outperforms the EM-based model in the transliteration tasks of four language pairs.

3.1 Background

3.1.1 Synchronous Context-Free Grammar

Synchronous Context-Free Grammar (SCFG) generalizes context-free grammar to generate strings concurrently in two languages (Lewis II and Stearns, 1968). Formally, a probabilistic SCFG is a tuple $\mathcal{G} = (\mathcal{N}, \mathcal{T}_s, \mathcal{T}_t, \mathcal{R}, \mathtt{S}, \boldsymbol{\Theta})$, where $\mathcal{N}$ is a set of nonterminal symbols, $\mathcal{T}_s$ and $\mathcal{T}_t$ are terminal symbols in the source side and target side respectively, $\mathcal{R}$ is a set of synchronous rewrite rules, $\mathtt{S} \in \mathcal{N}$ is the start symbol, and $\boldsymbol{\Theta}$ is the distribution of rule probabilities. The rules in SCFGs are in the form $\mathtt{A} \to \langle \beta \ / \ \gamma \ / \ \boldsymbol{a} \rangle$, where $\mathtt{A} \in \mathcal{N}$ is the parent nonterminal, $\beta \in (\mathcal{N} \cup \mathcal{T}_s)^*$ and $\gamma \in (\mathcal{N} \cup \mathcal{T}_t)^*$ are strings of terminals and nonterminals in the source and target languages respectively, and $\boldsymbol{a}$ is the one-to-one alignment between nonterminals in β and γ. Since we only discuss transliteration in this chapter, the nonterminals are always linked one-to-one from left to right without reordering, so we can omit the alignment and just write the rule as $\mathtt{A} \to \langle \beta \ / \ \gamma \rangle$. For each nonterminal $\mathtt{A} \in \mathcal{N}$, we denote $\mathcal{R}_\mathtt{A}$ as the set of rules with $\mathtt{A}$ as parent. The rule probabilities for each rule $r \in \mathcal{R}_\mathtt{A}$ must satisfy: $\sum_{r \in \mathcal{R}_\mathtt{A}} \theta_r = 1$.

To generate a string pair consisting of only terminals, we begin with the start symbol $\mathtt{S}$, then repeat applying rules to expand nonterminals on both sides, until the terminal string pair is generated. The whole generating process is named a *derivation*. The generating process forms a synchronous tree, in which leaf nodes corresponds to the terminal string pair, and internal nodes corresponds to nonterminal used in the derivation. The probability of a synchronous tree is the product of the probabilities of rules used in the derivation. Let $\mathcal{T}$ be a synchronous tree set, and f_r be the number of times that rule r is observed in $\mathcal{T}$, then the probability of $\mathcal{T}$ is

$$P(\mathcal{T}|\boldsymbol{\Theta}) = \prod_{\mathtt{A} \in \mathcal{N}} \mathrm{Multi}(\mathcal{T}|\boldsymbol{\theta}_A) = \prod_{\mathtt{A} \in \mathcal{N}} \prod_{r \in \mathcal{R}_\mathtt{A}} \theta_r^{f_r(\mathcal{T})} \tag{3.1}$$

where $\mathrm{Multi}(\mathcal{T}|\boldsymbol{\theta}_\mathtt{A})$ is the multinomial distribution for nonterminal $\mathtt{A}$.

The above probability model defines the probability of synchronous tree as the product of multinomial distributions with $\boldsymbol{\Theta}$ as parameters. In Bayesian learning, we could treat $\boldsymbol{\Theta}$ as random variables rather than parameters and define prior distributions on them. The conjugate prior of multinomial distribution is the Dirichlet distribution

$$\mathrm{Dir}(\boldsymbol{\theta}_A|\boldsymbol{\alpha}_A) = \frac{1}{\mathrm{Beta}(\boldsymbol{\alpha}_A)} \prod_{r\in\mathcal{R}_\mathcal{A}} \theta_r^{\alpha_r-1}, \tag{3.2}$$

where the concentration parameters $\boldsymbol{\alpha}$ control the shape of the Dirichlet distribution, and Beta is the multinomial Beta distribution defined as

$$\mathrm{Beta}(\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_1)\ldots\Gamma(\alpha_K)}{\Gamma(\sum_{k=1}^K \alpha_k)} \tag{3.3}$$

in which $\Gamma(x) = \int_0^\infty u^{x-1}e^{-u}\,\mathrm{d}u$ is the generalized factorial function[1]. In this Bayesian model, the posterior distribution on $\boldsymbol{\theta}$ is

$$P(\boldsymbol{\Theta}|\mathcal{T},\boldsymbol{\alpha}) \propto \prod_{\mathtt{A}\in\mathcal{N}} P(\mathcal{T}|\boldsymbol{\theta}_\mathtt{A}) P(\boldsymbol{\theta}_\mathtt{A}|\boldsymbol{\alpha}_\mathtt{A}) \propto \prod_{\mathtt{A}\in\mathcal{N}} \prod_{r\in\mathcal{R}_\mathtt{A}} \theta_r^{f_r(\mathcal{T})+\alpha_r-1} \tag{3.4}$$

which is the product of unnormalized Dirichlet distributions with parameter $(f_r(\mathcal{T}) + \boldsymbol{\alpha}_A)$. Thus we can write the posterior probability as product of Dirichlet distributions

$$P(\boldsymbol{\Theta}|\mathcal{T},\boldsymbol{\alpha}) = \prod_{\mathtt{A}\in\mathcal{N}} \mathrm{Dir}(\boldsymbol{\theta}_\mathtt{A}|\boldsymbol{f}_\mathtt{A}(\mathcal{T}) + \boldsymbol{\alpha}_\mathtt{A}) \tag{3.5}$$

The described SCFG models are parametric models since each of them has a fixed number of rules, each of which has a numerical parameter associated with it. For monolingual probabilistic context-free grammar (PCFG), there are two ways to construct nonparametric models: (1) Let the number of nonterminals grow unboundedly, as the infinite CFG models described in (Finkel et al., 2007; Liang et al., 2007); (2) Permit the number of rules to grow unboundedly, leading to adaptor grammars presented in (Johnson et al., 2007b). We follow the second one and extend it to synchronous adaptor grammar to model machine transliteration (see Section 3.2).

[1] For positive integers, $\Gamma(n) = (n-1)!$

3.1.2 Pitman-Yor Process

In probability theory, the *Pitman-Yor Process (PYP)* is a stochastic process that generates partitions of integers (Pitman and Yor, 1997). The Pitman-Yor process is often denoted as $\mathrm{PYP}(a, b, \mathcal{G}_0)$, where $a \in [0, 1]$ is a discount parameter, $b \geq 0$ is a concentration parameter, and $\mathcal{G}_0$ is the base distribution (or the *generator*, see follows).

The PYP is an extension of the Chinese Restaurant Process (CRP), so it is intuitive to describe the process using the restaurant metaphor. Assuming there are infinite number of round tables in the restaurant, each of which accommodates an infinite number of customers. Customers enter the restaurant sequentially and choose tables to sit around. Let z_i be the table number which the i^{th} customer chose. The first customer enters the restaurant and picks the first table, i.e. $z_1 = 1$. At a specific time, assuming there are already m tables which have been assigned with $n_1, \ldots, n_m$ customers sitting around respectively, and the total number of customers in the restaurant is n (i.e. $n = \sum_{k=1}^{m} n_k$), then the $(n+1)^{th}$ customer may choose an assigned table $k \in \{1, \ldots, m\}$ or choose a new table with number $(m+1)$ from the conditional distribution

$$P(z_{n+1}|z_1, \ldots, z_n) = \begin{cases} \frac{n_k - a}{n+b}, & \text{if } z_{n+1} = k \\ \frac{ma+b}{n+b}, & \text{if } z_{n+1} = m+1 \end{cases} \tag{3.6}$$

The joint probability of Pitman-Yor process for table indices is

$$\mathrm{PYP}(\boldsymbol{z}|a, b) = \frac{\prod_{k=1}^{m}(a(k-1)+b)\prod_{j=1}^{n_k-1}(j-a)}{\prod_{i=0}^{n-1}(i+b)}. \tag{3.7}$$

It is easy to verify that any permutation of $z_1, \ldots, z_n$ has the same probability in the Pitman-Yor process, so the Pitman-Yor process is *exchangeable*. This property results in efficient sampling procedure (see Section 3.2.2).

The above stochastic process generates sequences of integer table indices. If there is a value x_k (drawn from the base distribution $\mathcal{G}_0$) placed on the k^{th} table and customers yell out the value on the table which they choose, then we can obtain a value sequence $\boldsymbol{y}$ drawn from the Pitman-Yor process, with $y_i = x_{z_i}$ (for $i = 1, \ldots, n$).

In the first branch ($z_{n+1} = k$) in Equation (3.6), we can see that the table with more customers already sitting around (with big n_k) will be more likely be chosen again (with higher $P(z_{n+1} = k|z_1, \ldots, z_n)$). This demonstrates a kind of "rich get richer" dynamics, i.e. previous sampled values would be more likely sampled again in following sampling procedures. This dynamic is suitable for many machine learning tasks since they prefer sparse solutions to avoid the over-fitting problem.

Two special cases of Pitman-Yor process are interesting: (1) if $a = 1$, every customer would sit around a new table, so the values in sequence $\boldsymbol{y}$ are drawn independently from $\mathcal{G}_0$; (2) if $a = 0$, the Pitman-Yor process degenerates to the Dirichlet process (Pitman, 1995; Teh et al., 2006) with b as the concentration parameter. In this point of view, the Pitman-Yor process is an interpolation between the Dirichlet process and the base distribution. The discount parameter a provides more flexibility to control the tail behavior than the Dirichlet process. This makes the Pitman-Yor process useful for modeling data with power-law tails, e.g. word frequencies in natural language.

3.2 Synchronous Adaptor Grammars

3.2.1 Model

We extend the monolingual adaptor grammars to bilingual cases and propose the *Synchronous Adaptor Grammars* based on the Pitman-Yor process to learn bilingual tree structures. A Pitman-Yor Synchronous Adaptor Grammar (PYSAG) is a tuple $\mathcal{G} = (\mathcal{G}_s, \mathcal{N}_a, \boldsymbol{a}, \boldsymbol{b})$, where $\mathcal{G}_s = (\mathcal{N}, \mathcal{T}_s, \mathcal{T}_t, \mathcal{R}, \mathtt{S}, \boldsymbol{\Theta}, \boldsymbol{\alpha})$ is a Synchronous Context-Free Grammar (SCFG) (Lewis II and Stearns, 1968), $\mathcal{N}$ is a set of nonterminal symbols, $\mathcal{T}_s$ / $\mathcal{T}_t$ are source/target terminal symbols, $\mathcal{R}$ is a set of synchronous rewrite rules, $\mathtt{S} \in \mathcal{N}$ is the start symbol, $\boldsymbol{\Theta}$ is the distribution of rule probabilities, $\mathcal{N}_a \subseteq \mathcal{N}$ is the set of adapted nonterminals, $\boldsymbol{a} \in [0, 1], \boldsymbol{b} \geq 0$ are vectors of discount and concentration parameters both indexed by adapted nonterminals, and $\boldsymbol{\alpha}$ are Dirichlet prior parameters.

Algorithm 1 Generative Process of PYSAG

1: draw $\boldsymbol{\theta}_{\mathtt{A}} \sim \mathrm{Dir}(\alpha_{\mathtt{A}})$ for all $\mathtt{A} \in \mathcal{N}$
2: **for** each yield pair $\langle s \,/\, t \rangle$ **do**
3: SAMPLE(S) ▷ Sample from root
4: **return**

5: **function** SAMPLE(A) ▷ For $\mathtt{A} \in \mathcal{N}$
6: **if** $\mathtt{A} \in \mathcal{N}_a$ **then**
7: **return** SAMPLESAG(A)
8: **else**
9: **return** SAMPLESCFG(A)

10: **function** SAMPLESCFG(A) ▷ For $\mathtt{A} \notin \mathcal{N}_a$
11: draw synchronous rule $r = \langle \beta \,/\, \gamma \rangle \sim \mathrm{Multi}(\boldsymbol{\theta}_{\mathtt{A}})$
12: **for all** nonterminal $\mathtt{B} \in (\beta \cup \gamma)$ **do**
13: synchronous tree $t_{\mathtt{B}_i} \leftarrow$SAMPLE(B)
14: **return** BUILDTREE$(r, t_{\mathtt{B}_1}, t_{\mathtt{B}_2}, \ldots)$

15: **function** SAMPLESAG(A) ▷ For $\mathtt{A} \in \mathcal{N}_a$
16: draw cache index $z_{n+1} \sim P(z|z_{i<n})$, where
17: $$P(z|z_{i<n}) = \begin{cases} \frac{ma+b}{n+b}, & \text{if } z_{n+1} = m+1 \\ \frac{n_k - a}{n+b}, & \text{if } z_{n+1} = k \in \{1, \cdots, m\} \end{cases}$$
18: **if** $z_{n+1} = m+1$ **then** ▷ New entry
19: synchronous tree $t \leftarrow$ SAMPLESCFG(A)
20: $m \leftarrow m+1$ ▷ Update counts
21: $n_m = 1$ ▷ Update counts
22: INSERTTOCACHE$(\mathcal{C}_{\mathtt{A}}, t)$.
23: **else** ▷ Old entry
24: $n_k \leftarrow n_k + 1$
25: synchronous tree $t \leftarrow$ FINDINCACHE$(\mathcal{C}_{\mathtt{A}}, z_{n+1})$
26: **return** t

The generative process of a synchronous tree set $\mathcal{T}$ is described in Algorithm 1. First, rule probabilities are sampled for each nonterminal $\mathtt{A} \in \mathcal{N}$ (line 1) according to the Dirichlet distribution. Then synchronous trees are generated in the top-down fashion from the start symbol S (line 3) for each yield pair. For nonterminals that are not adapted, the grammar expands it just as the original synchronous grammar (function

SAMPLESCFG, line 10-14). For each adapted nonterminal $\mathtt{A} \in \mathcal{N}_a$, the grammar maintains a cache $\mathcal{C}_\mathtt{A}$ to store previously generated subtrees under $\mathtt{A}$. Let z_i be the subtree index in $\mathcal{C}_\mathtt{A}$, denoting the synchronous subtree generated at the i^{th} expansion of $\mathtt{A}$. At some particular time, assuming n subtrees rooted at $\mathtt{A}$ have been generated with m different types in the cache of $\mathtt{A}$, each of which has been generated for $n_1, \ldots, n_m$ times respectively[2]. Then the grammar either generates the $(n+1)^{th}$ synchronous subtree as SCFG (line 19) or chooses an existing subtree from the cache (line 25), according to the conditional probability $P(z|z_{i<n})$ defined in Equation (3.6). The process is shown in function SAMPLESAG, line 15-26.

The base distribution of the PYSAG is the Bayesian synchronous context-free grammar (described in Section 3.1.1). Since the rule probabilities $\boldsymbol{\Theta}$ in the Bayesian SCFG are used as hidden variables (sampled from hyperparameters $\boldsymbol{\alpha}$ and used to evaluate synchronous tree probabilities), we could integrate rule probabilities $\boldsymbol{\Theta}$ and directly obtain the joint probability of a particular sequence of synchronous trees:

$$P(\mathcal{T}|\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b}) = \prod_{\mathtt{A} \in \mathcal{N}} \frac{\mathrm{Beta}(\boldsymbol{\alpha}_\mathtt{A} + \boldsymbol{f}_\mathtt{A})}{\mathrm{Beta}(\boldsymbol{\alpha}_\mathtt{A})} \prod_{\mathtt{A} \in \mathcal{N}_a} \mathrm{PYP}(\boldsymbol{z}(\mathcal{T})|\boldsymbol{a}_\mathtt{A}, \boldsymbol{b}_\mathtt{A}) \tag{3.8}$$

where $\boldsymbol{f}_\mathtt{A}$ is the vector containing the number of times that rules $r \in \mathcal{R}_\mathtt{A}$ are used in the synchronous tree set, parameter $\boldsymbol{\alpha}_\mathtt{A}$ is the vector of Dirichlet hyperparameters for nonterminal $\mathtt{A}$, parameters $\boldsymbol{a}_\mathtt{A}$ and $\boldsymbol{b}_\mathtt{A}$ are vectors of discount and concentration parameters of the Pitman-Yor process, and $\boldsymbol{z}(\mathcal{T})$ are the indices of synchronous subtrees collected under adapted nonterminals.

The SAGs are synchronous extension of (monolingual) adaptor grammars (Johnson et al., 2007b). Differing from monolingual counterpart, the grammars and trees are both in the synchronous form in our model.

[2] Obviously, $n = \sum_{k=1}^{m} n_k$.

3.2.2 Inference

For the synchronous adaptor grammars based on the nonparametric Pitman-Yor process for machine transliteration, only raw name pairs are given, and we have to infer the hidden structure (synchronous trees) and estimate model parameters. As the caching nature of Pitman-Yor process, synchronous trees of different string pairs become depending on each other, so the joint probability of synchronous tree set can not be simply decomposed into the product of individual synchronous tree probabilities. Mathematically, given the set of string pairs $\boldsymbol{y} = \langle \boldsymbol{s} \,/\, \boldsymbol{t} \rangle$, the posterior distribution on $\mathcal{T}$ is

$$P(\mathcal{T}|\boldsymbol{y}) = \frac{P(\boldsymbol{y}|\mathcal{T})P(\mathcal{T})}{\sum_{\mathcal{T}'} P(\boldsymbol{y}|\mathcal{T}')P(\mathcal{T}')} \tag{3.9}$$

in which $P(\boldsymbol{y}|\mathcal{T}) = 1$ if $\text{yield}(\mathcal{T}) = \boldsymbol{y}$, and 0 otherwise, and $P(\mathcal{T})$ is the joint probability defined in Equation (3.8). Since synchronous trees of different string pairs are dependent on each other in PYSAGs, we have to enumerate all possible combinations to calculate the normalization constant in Equation (3.9), which is intractable. Fortunately, we are able to evaluate the (unnormalized) probability of a particular collection of synchronous trees, so we could estimate parameters of the Pitman-Yor processes using sampling techniques.

There is no obvious sampling method known to draw samples from Equation (3.8), so we extend the component-wise Metropolis-Hastings algorithm (Johnson et al., 2007b) to the synchronous case. Let $\mathcal{T}_{-i}$ be the set of sampled trees except the i^{th} one. As the Pitman-Yor process is exchangeable, we can always treat the i^{th} sample as the final sample after sampling $\mathcal{T}_{-i}$. In the Metropolis-Hastings sampling algorithm for PYSAG, we draw the synchronous tree t_i' from some proposal distribution $Q(t_i|y_i, \mathcal{T}_{-i})$, then accept the new sample t_i' with probability

$$\begin{aligned} A(t_i, t_i') &= \min\left\{1, \frac{P(\mathcal{T}'|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})\; Q(t_i|y_i, \mathcal{T}_{-i})}{P(\mathcal{T}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})\; Q(t_i'|y_i, \mathcal{T}_{-i})}\right\} \\ &= \min\left\{1, \frac{P(t_i'|y_i, \mathcal{T}_{-i}, \boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})\; Q(t_i|y_i, \mathcal{T}_{-i})}{P(t_i|y_i, \mathcal{T}_{-i}, \boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})\; Q(t_i'|y_i, \mathcal{T}_{-i})}\right\} \end{aligned} \tag{3.10}$$

where $\mathcal{T}' = \{t'_i\} \cup \mathcal{T}_{-i}$ and $\mathcal{T} = \{t_i\} \cup \mathcal{T}_{-i}$ represents the whole synchronous tree set including the new sample and old sample respectively. The art of Metropolis-Hastings algorithm is that the acceptance probability is a ratio of probabilities, so the evaluation difficulty of normalization becomes a common factor between the probabilities of both old samples and new samples, which can be cancelled.

In theory, the proposal distribution Q could be any distribution if it never assigns zero probability. In practice, the proposal distribution Q should be close enough to the true distribution P to avoid high rejection rate. In monolingual adaptor grammars, Johnson et al. (2007b) use the PCFG Approximation as the proposal distribution and report very small rejecting rate. However, this proposal modifies the SCFG grammars in each sampling step, so the parse forest has to be reconstructed each time. This parsing step is time-consuming, especially in the synchronous situation[3]. Therefore, we do not implement the PCFG approximation method for synchronous adaptor grammar due to efficiency reason. Instead, we choose the probabilistic SCFG as the proposal distribution (similar to the PCFG (Johnson et al., 2007a)). During inference, we collect statistics of rules as well as the subtrees rooted at adapted nonterminals. One instance is considered at a time. To draw a tree from yield pair $y_i = \langle s_i \ / \ t_i \rangle$, we exclude the counts of its rule usage and then estimate the probability of rule $r \in \mathcal{R}_{\mathtt{A}}$ in Q by relative frequency[4]

$$\theta_r = \frac{[f_r]_{-i} + \alpha_r}{\sum_{r' \in \mathcal{R}_{\mathtt{A}}} [f_{r'}]_{-i} + \alpha_{r'}} \tag{3.11}$$

where $\mathcal{R}_{\mathtt{A}}$ is the set of rules rooted at $\mathtt{A}$, and $[f_r]_{-i}$ is the number of times that rule r is used in the tree set $\mathcal{T}_{-i}$. We pre-parse the training instances before inference and save the structure of synchronous parse forests. During the inference, we only change rule probabilities in parse forests without changing the forest structures. We use the sampling algorithm described in (Blunsom and Osborne, 2008) to draw a synchronous tree from the parse forest according to the proposal Q.

[3] We implement the synchronous CKY-like parsing algorithm (Wu, 1997), with $O(|s|^3 \ |t|^3)$ complexity.
[4] There is a typo in our original paper (Huang et al., 2011), which is fixed here.

3.3 Machine Transliteration

3.3.1 Grammars

To verify the usefulness of the proposed synchronous adaptor grammars, we conduct experiments on the machine transliteration task. We demonstrate how machine transliteration could be modeled as the synchronous adaptor grammars in this section.

For machine transliteration, we use an adapted nonterminal Syl to capture the syllable equivalents between two languages. There may be multiple characters on both the source and target sides in a syllable. One possible way to model the many-to-many syllable mappings is to enumerate all possible subsequence pairs on the source and target sides. However, assuming the source name has $|s|$ characters and the target name has $|t|$ characters, the number of rules has $O(|s|^2\ |t|^2)$ complexity, which is large especially for long name pairs.

To reduce the grammar size, we use an alternative representation, in which we restrict the leftmost characters on both sides to be aligned one-by-one and introduce a special empty character ε to link unaligned characters. For instance, we do not directly allow Syl $\rightarrow \langle$a a l l / 阿 尔$\rangle$ or Syl $\rightarrow \langle$x / 克 斯$\rangle$. Instead, we link the bilingual characters (including empty ones) in sequence from left to right, such as Syl $\xrightarrow{*} \langle$a a l l / 阿 尔 ε $\varepsilon\rangle$ and Syl $\xrightarrow{*} \langle$x ε / 克 斯$\rangle$. In addition, we use nonterminal NEC to represent single character pair without any empty character (e.g. $\langle$a / 阿$\rangle$), nonterminal SEC represents single character pair of empty source and non-empty target (e.g. $\langle\varepsilon$ / 斯$\rangle$), and nonterminal TEC represents single character pair of non-empty source and empty target (e.g. $\langle$l / $\varepsilon\rangle$). We also use three nonterminals NECs, SECs and TECs to represent corresponding pairs of one or more characters, e.g. NECs $\xrightarrow{*} \langle$a a / 阿 尔$\rangle$, SECs $\xrightarrow{*} \langle\varepsilon$ / 斯$\rangle$, and TECs $\xrightarrow{*} \langle$l l / ε $\varepsilon\rangle$. Although the above design introduces some useless character pairs, our goal is to learn the syllable equivalents which are captured by the adaptor Syl, so we are not interested in the subtree structure inside syllables.

In detail, we design following grammar[5] to learn syllable mappings:

$$\begin{aligned}
\texttt{Name} &\rightarrow \langle \underline{\texttt{Syl}} \;/\; \underline{\texttt{Syl}} \rangle^{+} \\
\underline{\texttt{Syl}} &\rightarrow \langle \texttt{NECs} \;/\; \texttt{NECs} \rangle && (3.12) \\
\underline{\texttt{Syl}} &\rightarrow \langle \texttt{NECs SECs} \;/\; \texttt{NECs SECs} \rangle && (3.13) \\
\underline{\texttt{Syl}} &\rightarrow \langle \texttt{NECs TECs} \;/\; \texttt{NECs TECs} \rangle && (3.14) \\
\texttt{NECs} &\rightarrow \langle \texttt{NEC} \;/\; \texttt{NEC} \rangle^{+} \\
\texttt{SECs} &\rightarrow \langle \texttt{SEC} \;/\; \texttt{SEC} \rangle^{+} \\
\texttt{TECs} &\rightarrow \langle \texttt{TEC} \;/\; \texttt{TEC} \rangle^{+} \\
\texttt{NEC} &\rightarrow \langle s_i \;/\; t_j \rangle \\
\texttt{SEC} &\rightarrow \langle \varepsilon \;/\; t_j \rangle \\
\texttt{TEC} &\rightarrow \langle s_i \;/\; \varepsilon \rangle
\end{aligned}$$

where the start symbol Name represents the transliteration name pair, the adapted nonterminal Syl may be expanded to the pair of syllables with the same length (rule 3.12), with less source length (rule 3.13), or with less target length (rule 3.14), and s_i and t_j enumerate over the source and target character set respectively. We refer this grammar as the *syllable grammar*. Figure 3.1 shows an example for the English-Chinese transliteration.

The above syllable grammar is able to learn inner-syllable dependencies. However, the selection of the target characters also depend on the context. For example, the following three instances are found in the training set:

$$\begin{aligned}
&\langle \texttt{a a b y e} \;/\; \texttt{奥[ao] 比[bi]} \rangle \\
&\langle \texttt{a a g a a r d} \;/\; \texttt{埃[ai] 格[ge] 德[de]} \rangle \\
&\langle \texttt{a a l t o} \;/\; \texttt{阿[a] 尔[er] 托[tuo]} \rangle
\end{aligned}$$

where the same English syllable $\langle$a a$\rangle$ are transliterated to $\langle$奥[ao]$\rangle$, $\langle$埃[ai]$\rangle$ and

[5] Similar to (Johnson, 2008), the adapted nonterminal are underlined. Similarly, we also use rules in the regular expression style $\texttt{X} \rightarrow \langle \texttt{A} \;/\; \texttt{A} \rangle^{+}$ to denote the following three rules:

$$\begin{aligned}
\texttt{X} &\rightarrow \langle \texttt{As} \;/\; \texttt{As} \rangle \\
\texttt{As} &\rightarrow \langle \texttt{A} \;/\; \texttt{A} \rangle \\
\texttt{As} &\rightarrow \langle \texttt{A As} \;/\; \texttt{A As} \rangle
\end{aligned}$$

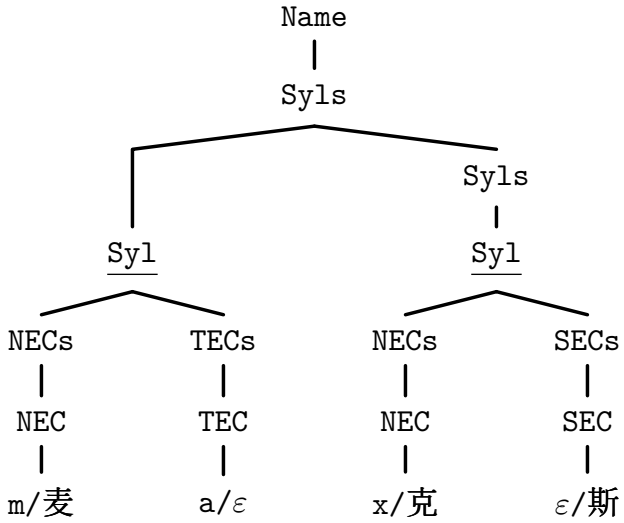


Figure 3.1: A parse tree of syllable grammar for `En-Ch` transliteration.

⟨阿`[a]`⟩ respectively, depending on the following syllables. To model these contextual dependencies, we propose the hierarchical synchronous adaptor grammar. The two-layer *word grammar* is obtained by adding following rules:

$$\texttt{Name} \rightarrow \langle \underline{\texttt{Word}} \,/\, \underline{\texttt{Word}} \rangle^{+}$$
$$\underline{\texttt{Word}} \rightarrow \langle \underline{\texttt{Syl}} \,/\, \underline{\texttt{Syl}} \rangle^{+}$$

where a new adapted nonterminal $\underline{\texttt{Word}}$ is introduced to capture the inter-syllable dependencies. Figure 3.2 shows an example for the English-Japanese transliteration, where the syllable combinations between English transcript and Japanese Katakana are captured by the adapted nonterminal $\underline{\texttt{Word}}$ (e.g. ⟨`s e n` / セ ン⟩).

Following (Johnson, 2008), we might further add a new adapted nonterminal $\underline{\texttt{Col}}$ to learn the word collocations. The following rules appear in the *collocation grammar*:

$$\texttt{Name} \rightarrow \langle \underline{\texttt{Col}} \,/\, \underline{\texttt{Col}} \rangle^{+}$$
$$\underline{\texttt{Col}} \rightarrow \langle \underline{\texttt{Word}} \,/\, \underline{\texttt{Word}} \rangle^{+}$$
$$\underline{\texttt{Word}} \rightarrow \langle \underline{\texttt{Syl}} \,/\, \underline{\texttt{Syl}} \rangle^{+}$$

Figure 3.3 shows a synchronous tree example of the collocation grammar, where the whole name is captured by the adapted nonterminal $\underline{\texttt{Col}}$.

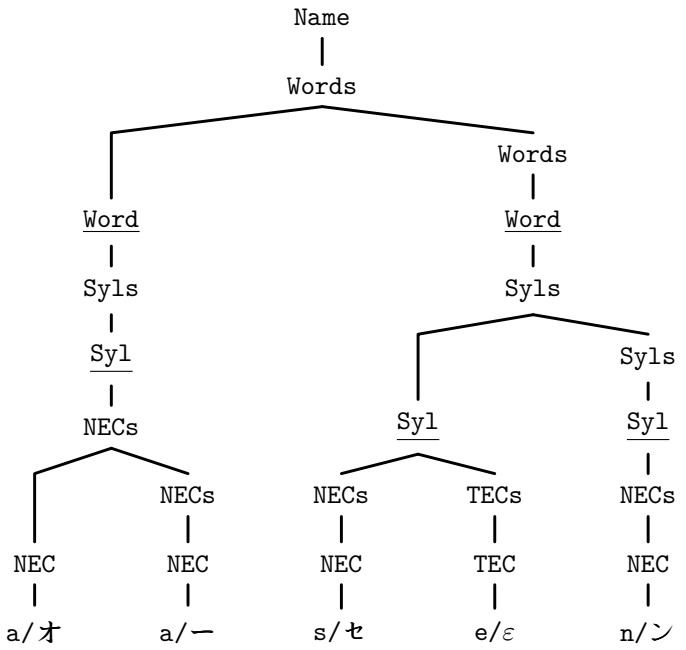


Figure 3.2: A parse tree of word grammar for `En-Ja` transliteration.

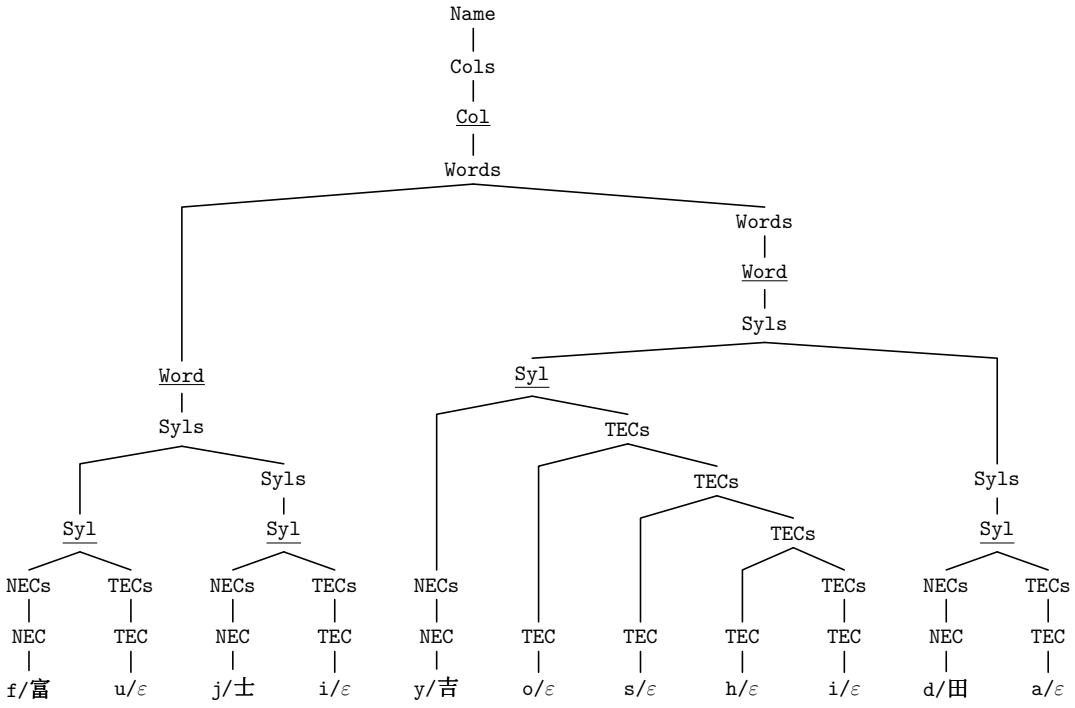


Figure 3.3: A parse tree of collocation grammar for `Jn-Jk` transliteration.

3.3.2 Transliteration Model

We use the n-gram translation model (Li et al., 2004) as the transliteration model in our experiments. Denote the bilingual pair as $y = \langle s \ / \ t \rangle$, and it could be split into bilingual syllable sequences $(y_1, \ldots, y_K) = (\langle s_1 \ / \ t_1 \rangle, \ldots, \langle s_K \ / \ t_K \rangle)$. This transliteration model factorizes the probability of $P(y)$ into n-gram probabilities

$$P(y) = P(y_1^K) = \prod_{k=1}^{K} P(y_k | y_1^{k-1}) \approx \prod_{k=1}^{K} P(y_k | y_{k-n+1}^{k-1}). \tag{3.15}$$

After the inference step for synchronous adaptor grammar described in Section 3.2.2, we construct joint segmentation lattice for each training instance. We first generate a merged grammar G' using collected subtrees under adapted nonterminals, then use synchronous parsing to obtain probabilities in the segmentation lattice. Specifically, we *flatten* the collected subtrees under Syl, i.e. removing internal nodes, to construct new synchronous rules. For example, we could get two rules from the tree in Figure 3.1:

$$\texttt{Syl} \rightarrow \langle \texttt{m a} \ / \ 麦 \rangle$$
$$\texttt{Syl} \rightarrow \langle \texttt{x} \ / \ 克 \ 斯 \rangle$$

If multiple subtrees are flattened to the same synchronous rule, we sum up the counts of these subtrees. For rules with non-adapted nonterminal as parent, we assign the probability as the same of the sampled rule probability, i.e. let $\theta'_r = \theta_r$. For the adapted nonterminal Syl, there are two kinds of rules: (1) the rules in the original probabilistic SCFG, and (2) the rules flattened from subtrees. We assign the rule probability as

$$\theta'_r = \begin{cases} \frac{ma+b}{n+b} \cdot \theta_r, & \text{if } r \text{ is original SCFG rule} \\ \frac{n_r - a}{n+b}, & \text{if } r \text{ is flatten from subtree} \end{cases} \tag{3.16}$$

where a and b are the parameters associated with Syl, m is the number of types of different rules flatten from subtrees, n_r is the count of rule r, and n is the total number of flatten rules. One may verify that the rule probabilities are well normalized. Based on this merged grammar G', we parse the training string pairs, then encode the

parsed forest into the lattice. Figure 3.4 show a lattice example for the string pair ⟨a a l t o / 阿[a] 尔[er] 托[tuo]⟩. The transition probabilities in the lattice are the inside probabilities of corresponding Syl node in the parse forest. After building the segmentation lattice, we train language model for bilingual syllables from the lattice.

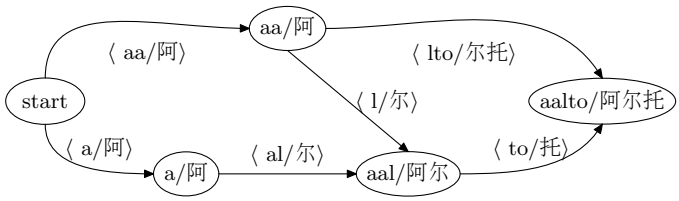


Figure 3.4: An example of decoding lattice for SAG.

In transliteration step, given the source string s and grammar $\mathcal{G}$, we want to find a translation $\hat{t}$ that maximizes the conditional probability:

$$\begin{aligned}\hat{t} &= \arg\max_t P(t|s,\mathcal{G}) \\ &= \arg\max_t \frac{P(s,t|\mathcal{G})}{P(s|\mathcal{G})} \\ &= \arg\max_t P(s,t|\mathcal{G}) \\ &= \arg\max_t \sum_d \delta(s,t|d,\mathcal{G})P(d|\mathcal{G}) \end{aligned} \tag{3.17}$$

where $P(d|\mathcal{G})$ is the probability of derivation d under grammar $\mathcal{G}$, and $\delta(s,t|d,\mathcal{G}) = 1$ if the yield pair of d is $\langle s,t\rangle$ and $\delta(s,t|d,\mathcal{G}) = 0$ otherwise. However, there are exponential number of derivations, so the above optimal *translation decoding* is often approximated by the optimal *derivation decoding*, i.e. we find the derivation $\hat{d}$ in

$$\begin{aligned}\hat{d} &= \arg\max_d P(d|\mathcal{G}) \\ s.t. \quad & \text{yield}(d) = \langle s,-\rangle \end{aligned} \tag{3.18}$$

We use the Viterbi algorithm with beam search (Li et al., 2004) to find the best derivation d instead of summing up (exponential number of) derivations.

3.4 Experiments

3.4.1 Data and Settings

We conduct experiments on the following four language pairs from the ACL Named Entities Workshop (NEWS 2009) datasets[6]:

`En-Ch`: English name to Chinese name;

`En-Ja`: English name to Japanese Katakana;

`En-Ko`: English name to Korean Hangul;

`Jn-Jk`: Japanese name (written in English) to Japanese Kanji.

Lang	Data	#Entry	#Src Char	#Tgt Char	#Tgt Voc
`En-Ch`	Train	31961	218073	101205	374
	Dev	2896	19755	9160	
	Test	2896	19864	9246	
`En-Ja`	Train	27993	188941	131275	81
	Dev	1818	12105	8358	
	Test	1788	11961	8293	
`En-Ko`	Train	4840	32150	15128	713
	Dev	998	6656	3134	
	Test	993	6606	3121	
`Jn-Jk`	Train	16352	105916	34231	1600
	Dev	3539	23248	7462	
	Test	3868	25668	8020	

Table 3.1: Transliteration data statistics.

In the data preparation step, we remove spaces and the apostrophe symbol (`'`) within names. For example, the Japanese name "`Kan'ichi`" would be converted to "`Kanichi`". These removals confuse syllable boundaries and may hurt the performance. Note that this preprocessing step is the same for both the baseline model and proposed model, so they

[6] http://www.acl-ijcnlp-2009.org/workshops/NEWS2009/

are used in experiments for simplicity. Table 3.1 gives some statistics of the datasets. For En-Ch, there is only one Chinese reference per English name; while for other language pairs, there may be multiple references for the same English name.

We implement the joint source-channel model (Li et al., 2004) as the baseline system, in which the orthographic syllable alignment is automatically derived by the Expectation-Maximization (EM) algorithm. Since EM tends to memorize the training instance as a whole, Li et al. (2004) restrict the target side to be single character in syllable equivalents for English-Chinese experiments. We follow their work and apply the single-character restriction to other language pairs.

Our method can be viewed as the Bayesian extension of the EM-based baseline. Since PYSAGs could learn accurate and compact transliteration units, we do not need the single-character restriction any more. In the inference step of PYSAGs, we first run the sampler through the whole training corpus for 10 iterations (burn-in), then collect adapted subtree statistics for every 10 iterations, and finally stop after 20 collections.

In general, we have no idea which values should be assigned to the hyperparameters $\boldsymbol{a}$ and $\boldsymbol{b}$. Following (Johnson and Goldwater, 2009), we put a $\mathrm{Beta}(\alpha, \beta)$ prior on $\boldsymbol{a}$ and a "vague" $\mathrm{Gamma}(10, 0.1)$ prior on $\boldsymbol{b}$ to model the uncertainty of hyperparameters. We tune α and β from $\{0.3, 1, 3\}$, and choose the parameters with highest word accuracy on the development set. After each iteration, we resample each of hyperparameters from the posterior distribution of hyperparameters using a slice sampler (Neal, 2003). We modify the open-source implementation of slice sampler provided by Mark Johnson[7].

For both the baseline model and our proposed models, we build the segmentation lattice after training. Then we train a 3-order language model with the Witten-Bell smoothing (Witten and Bell, 1991) from the lattice using the SRI language model toolkit[8]. After that, the Viterbi algorithm with beam search (Li et al., 2004) is used in decoding for both the EM-based model and the proposed Bayesian models.

[7] http://web.science.mq.edu.au/~mjohnson/Software.htm
[8] http://www.speech.sri.com/projects/srilm/

3.4.2 Evaluation Metrics

For evaluation, we report the standard evaluation metrics defined in (Li et al., 2009a), and report the *word accuracy* and *mean F-score* metrics. Following notations used in (Li et al., 2009a), we denote:

N as the total number of names (source words) in the test set;

n_i as the number of reference transliterations for i^{th} name;

$r_{i,j}$ as the j^{th} reference transliteration for the i^{th} name;

$c_{i,k}$ as the k^{th} candidate transliteration output by transliteration system for i^{th} name. Then the word accuracy and mean F-score metrics can be defined as follows:

- **Word Accuracy in Top-1 (Acc)**

  The word accuracy is also known as the word error rate, it measures the correctness of the first transliteration candidate in the candidate list produced by a transliteration system. If the first transliteration candidate matches at least one of the references, the result is considered correct. Otherwise, if the first transliteration candidate matches none of the references, the transliteration is measured as the wrong one. The overall word accuracy is defined as the correct word percentage over the whole test set:

$$Acc = \frac{1}{N}\sum_{i=1}^{N}\left\{\begin{array}{ll} 1 & \text{if } \exists r_{i,j} : r_{i,j} = c_{i,1}; \\ 0 & \text{otherwise} \end{array}\right\} \tag{3.19}$$

- **Fuzziness in Top-1 (Mean F-score F_1)**

  The mean F-score measures how different the top transliteration candidate is from its closest reference. First, the Edit Distance (ED) and the Longest Common Subsequence (LCS) between the candidate word and each reference are calculated using dynamic programming. The edit distance measures the minimum number of single-character editing operations, including insertions and deletions (no replacements in calculation). For example, the edit distance between "`abcdef`" and

"afcdg" is 5, including deletions of b, e, f from the first string, and insertions of f, g into the second string. The longest common subsequence is defined as the longest subsequence (not substring) common in the two sequences. For example, the longest common subsequence between "abcdef" and "afcdg" is "acd". Then for each name in the test set, we define the best matching reference as the reference with the minimal edit distance from the candidate:

$$r_{i,m} = \arg\min_{j} ED(c_{i,1}, r_{i,j}) \tag{3.20}$$

Finally, the best matching reference is used to calculated the Precision (P), the Recall (R) and their harmonic mean (F_1) for the i^{th} word:

$$P = \frac{LCS(c_{i,1}, r_{i,m})}{|c_{i,1}|} \tag{3.21}$$

$$R = \frac{LCS(c_{i,1}, r_{i,m})}{|r_{i,m}|} \tag{3.22}$$

$$F_1 = \frac{2\,P\,R}{P + R} \tag{3.23}$$

The overall mean F-score is the average F-score over the whole test set.

The above two metrics are both defined over the top-1 candidates. One may argue that multiple grapheme may have the same pronunciations in transliteration. Actually, Li et al. (2009a) also define other metrics to evaluate other transliterated name in the candidate list, such as the MAP_{10}. However, according to some national standards (e.g. *The Chinese Phonetic Alphabet Spelling Rules for Chinese Names*[9]), there are usually only one grapheme representation is considered correct. As a result, we only adopt the above two metrics to evaluate top-1 candidates. The evaluation script can be downloaded from the website of the NEWS 2009[10].

[9] http://www.njqb.gov.cn/qwdt/ggl/201209/W020120911597707484864.pdf
[10] https://translit.i2r.a-star.edu.sg/news2009/evaluation/

3.4.3 Results

Language	Model	Dev(%)	Test(%)
En-Ch	(Li et al., 2004)	66.8 / 87.1	66.8 / 86.7
	Syl	66.6 / 87.0	66.6 / 86.6
	Word	67.1 / 87.2	67.0 / 86.7
	Col	**67.2 / 87.1**	**66.9 / 86.7**
En-Ja	(Li et al., 2004)	43.7 / 82.2	44.7 / 82.2
	Syl	43.7 / 81.8	44.9 / 82.4
	Word	**44.0 / 82.5**	**45.9 / 82.6**
	Col	44.0 / 81.8	44.5 / 82.2
En-Ko	(Li et al., 2004)	28.1 / 63.1	27.7 / 63.3
	Syl	33.6 / 66.8	32.0 / 65.4
	Word	**33.9 / 66.2**	**34.0 / 65.6**
	Col	33.8 / 66.1	33.9 / 66.0
Jn-Jk	(Li et al., 2004)	57.5 / 73.3	58.5 / 73.7
	Syl	60.7 / 75.5	61.7 / 75.9
	Word	60.5 / 75.4	61.5 / 75.8
	Col	**60.9 / 75.5**	**61.7 / 76.1**

Table 3.2: Transliteration results, in the format of *word accuracy / mean F-score*. "Syl","Word" and "Col" denote the syllable, word and collocation grammar respectively.

Table 3.2 presents the transliteration results of all experiments. From this table, we draw following conclusions:

1. The proposes Bayesian models achieve better performance or at least comparable performance than the baseline EM-based model on both the development set and the test set for all language pairs. We conclude that the PYSAGs could find good syllable mappings from the raw name pairs without any heuristics or restrictions. In this point of view, the proposed method is language independent.

2. If we sort the improvements on the test set ($Acc\%$) from the highest to the lowest, we can get: `En-Ko`(6.3) > `Jn-Jk`(3.2) > `En-Ja`(1.2) > `En-Ch`(0.1). We also observe from Table 3.1 that the number of training instances are exactly in the reversed order: `En-Ko`($4.8K$) < `Jn-Jk`($16K$) < `En-Ja`($28K$) < `En-Ch`($32K$). These facts

may be explained that the prior knowledge play a more important role for small data than large-scale data. For sufficient large-scale data, we can just let "data speak themselves", since EM could already learn good syllable alignments.

3. Comparing among the best (absolute) accuracy values for different language pairs, we have: `En-Ch`(67.0) > `Jn-Jk`(61.7) > `En-Ja`(45.9) > `En-Ko`(34.0). In general, higher performance could be achieved with more training data. One exception is that the result of `Jn-Jk` (with smaller training set) is higher than `En-Ja`. The reason might because Japanese Kanji has relative small (maybe fixed) set of English correspondences and it is easy to split the source English name into syllable parts. For example, "`chiyako/`千夜子" can be easily split into "`chi-ya-ko/`千-夜-子" without ambiguity. To transliterate from western names to Japanese, however, there may be difficult to find the corresponding Katakana in Japanese.

4. The word and collocation grammars achieve slightly better performance than the syllable grammars, although the improvements are not significant. These facts do not give strong evidences to support the assumption that the context information are helpful. We guess the reason is that the instances in transliteration are very short, so syllable grammars are good enough while the word and collocation rules become very sparse, which results in unreliable probability estimation.

For the `En-Ch` experiments, the only syllable pair that violates the single-character restriction is ⟨`x` / 克 斯⟩. We perform additional `En-Ch` baseline experiments by replacing the single English character ⟨`x`⟩ with two characters ⟨`K S`⟩ and run the baseline experiments. The results of replacement have been reported in our previous work (Huang et al., 2011) as {Dev 67.8/86.9}, which improve the baseline results {Dev (66.8/87.1)} in Table 3.2. We can conclude that the single-character restriction hurts the performance. Furthermore, for other language pairs, there may not exist simple replacements. Compared with EM, the proposed PYSAGs automatically learn syllable equivalents without restrictions and achieve better performance.

3.4.4 Discussion

We examine the learned syllable mappings in PYSAGs. Table 3.3 shows En-Ch examples of learned syllable equivalents with largest collected counts in the final sampled tree of the syllable grammar. As comparison, Table 3.4 shows the syllable equivalents collected from the 1-best output of the baseline EM algorithm.

s/斯[si]/1669	ro/罗[luo]/531	la/拉[la]/382
t/特[te]/728	son/森[sen]/442	tt/特[te]/380
man/曼[man]/703	k/克[ke]/408	l/尔[er]/367
d/德[de]/579	ma/马[ma]/390	ton/顿[dun]/360
ck/克[ke]/564	co/科[ke]/387	ri/里[li]/342
de/德[de]/564	ll/尔[er]/383	ra/拉[la]/339
x/克[ke] 斯[si]/40	x/克[ke]/3	x/斯[si]/1

Table 3.3: Examples of sampled En-Ch syllable mappings (total 79141, type 6880) in the final sampled tree. Chinese Pinyin (in square brackets) and the counts of syllable equivalents are given.

s/斯[si]/6186	ri/里[li]/1114	ll/尔[er]/924
l/尔[er]/3172	ro/罗[luo]/1093	p/普[pu]/841
t/特[te]/2434	c/克[ke]/1062	m/姆[mu]/800
d/德[de]/2355	k/克[ke]/1048	ra/拉[la]/759
g/格[ge]/1582	ck/克[ke]/971	le/尔[er]/750
b/布[bu]/1497	man/曼[man]/933	de/德[de]/718
x/克[ke] 斯[si]/0	x/克[ke]/90	x/斯[si]/139

Table 3.4: Examples of learned En-Ch syllable mappings (total 101205, type 5466) in the 1-best alignment output by EM baseline. Chinese Pinyin (in square brackets) and the counts of syllable equivalents are given.

From these tables, we can see that the PYSAGs and baseline model find slightly different syllable mappings from raw name pairs. Note that the EM baseline restricts only one character in the Chinese side, while PYSAGs do not have any heuristics or restrictions. Specifically, we are interested in the English token $\langle \mathrm{x} \rangle$, which is the only one that

has two corresponding Chinese characters ⟨克[ke] 斯[si]⟩. Table 3.3 demonstrates that many of these correct mappings are discovered by PYSAGs, while these equivalents can not be found if we restrict the Chinese side to be only one character (Li et al., 2004), as shown in Table 3.4.

Another interesting example is the Japanese Katakana symbol "ー", which is used to indicate the preceding vowel is a long vowel. As the original joint source-channel model restricts that the Japanese side of syllable mappings to be a single Katakana, the symbol "ー" has many correspondences in English, such as ⟨r⟩, ⟨er⟩, ⟨e⟩, ⟨a⟩, ⟨o⟩, ⟨y⟩, depending on the previous syllables. In contrast, the proposed SAG model could recognize the symbol "ー" should not be split from its previous syllable, and learn many-to-many syllable mappings. Some learned examples are shown as follows:

⟨ner/ナ[na] ー[-]⟩ ⟨ley/リ[li] ー[-]⟩ ⟨mar/マ[ma] ー[-]⟩

Besides the above unbreakable syllable mappings, our PYSAG model could also learn big breakable syllable equivalents. For example, the following syllable equivalents (with separated form) can be found in sampled trees for several times:

⟨ski/斯[si] 基[ji]⟩ ⇒ ⟨s/斯[si]⟩ ⟨ki/基[ji]⟩

⟨mc/麦[mai] 克[ke]⟩ ⇒ ⟨m/麦[mai]⟩ ⟨c/克[ke]⟩

⟨man/マ[ma] ン[n]⟩ ⇒ ⟨ma/マ[ma]⟩ ⟨n/ン[n]⟩

⟨ber/ベ[be] ル[ru]⟩ ⇒ ⟨be/ベ[be]⟩ ⟨r/ル[ru]⟩

In general, these big syllable equivalents may be separated into small syllable mappings. They are considered as a whole since the PYSAGs give higher probabilities to the whole syllable equivalents than the separated ones due to their high-frequency appearance. This observation explains why the PYSAG sampled less syllable equivalents in total (with more types) in Table 3.3 than those equivalents learned by EM in Table 3.4 (with less types), and the frequencies of sampled syllable equivalents is smaller than the corresponding ones learned by EM. Similar results could be found for other language pairs.

3.5 Summary

In this chapter, we propose synchronous adaptor grammars for machine transliteration. Based on the sampling, the PYSAGs could automatically discover syllable equivalents without any heuristics or restrictions. In this point of view, the proposed model is language independent. The joint source-channel model is then used for training and decoding. Experimental results on the transliteration tasks of four language pairs show that the proposed method outperforms the EM-based baseline system. We also compare grammars in different layers and find that the two-layer grammars are suitable for the transliteration task, although the performance difference between grammar layers are not significant.

Chapter 4

Feature-based Constituent-Context Model

The basic Constituent Context Model (CCM) (Klein and Manning, 2002) has been described in Section 2.2.1. Although CCM achieves promising results in short sentences, its performance drops for longer sentences. In this chapter, we propose a general feature-based framework for CCM in which various overlapping features could be easily added. Features take the log-linear form with local normalization, where we can still use the EM algorithm to estimate model parameters with minor change in the maximization step. To avoid overfitting, we use ℓ_1-norm regularization to control the model complexity. Furthermore, previous induction models (Klein and Manning, 2002; Smith and Eisner, 2004; Mirroshandel and Ghassem-Sani, 2008; Golland et al., 2012) train and evaluate models on the same dataset, so there is no reasonable way to choose model parameters. We advocate using a separated validation set to perform model selection, and measure the trained model on additional test set. Under this framework, we could automatically choose suitable model parameters instead of setting them empirically. We also examine the sparse model issues in this chapter.

4.1 Feature-based CCM

4.1.1 Model Definition

Motivated by (Berg-Kirkpatrick et al., 2010; Li et al., 2012), the basic idea behind the feature-based CCM is to factorize the multinomial distribution over sequences into small factors that describe overlapped aspects of constituents and distituents (a.k.a. non-constituents).

Formally, let B be a boolean matrix with entries indicating whether the corresponding span encloses constituent or distituent. As explained in Section 2.2.1, some bracketing B may not corresponds to parse tree. We just ignore those bracketings in probability calculation, i.e. let $P(B) = 0$. We denote $\mathcal{B}_T$ as the set of bracketings with tree representations. For tree-equivalent bracketing $B \in \mathcal{B}_T$, denote T_B as the corresponding tree representation. We define factors in the log-linear form with local normalization. Let $F_k(k = 1, \ldots, K)$ be K different factors. Each factor F_k corresponds to a n_k-dimensional feature vector $\boldsymbol{f}_k$. For each feature vector, there is a n_k-dimensional weight vector $\boldsymbol{w}_k$ measuring the importance for each dimension. Note for the k^{th} factor F_k, the corresponding multinomial parameter in traditional CCM is now treated as a function of weights $\boldsymbol{w}_k$. Using these notations, we define the log-linear factor F_k for span $\langle i, j \rangle$ in some bracketing B for sentence S as

$$\begin{aligned} F_k(S_{\langle i,j \rangle} | \boldsymbol{w}_k) &= P_k(S_{\langle i,j \rangle} | B_{\langle i,j \rangle}, \boldsymbol{w}_k) \\ &= \frac{\exp(\boldsymbol{w}_k \cdot \boldsymbol{f}_k(S_{\langle i,j \rangle}))}{\sum_{\boldsymbol{v}} \exp(\boldsymbol{w}_k \cdot \boldsymbol{f}_k(\boldsymbol{v}))} \end{aligned} \tag{4.1}$$

where $\boldsymbol{f}_k$ returns a feature vector, $\boldsymbol{w}_k$ is the corresponding weight vector, and $(\cdot)$ denotes the inner product of vectors. The denominator sums over the unnormalized probabilities (as defined in the numerator) for all possible factor values of $\boldsymbol{f}_k(\boldsymbol{v})$. Since there are exponential values $\boldsymbol{v}$ respect to the dimension number, we approximately calculate this summation only over values that appear in the training corpus.

Similar to CCM, there are constituent factors and distituent factors in the feature-based model. Constituent and distituent factors only define probabilities over constituent and distituent spans respectively. To distinguish constituent and distituent factors, we define a factor category function δ_k as

$$\delta_k = \begin{cases} +1, & \text{if } F_k \text{ is constituent factor} \\ -1, & \text{if } F_k \text{ is distituent factor} \end{cases} \tag{4.2}$$

Then the joint probability of $P(S, B|\boldsymbol{w})$ can be defined:

$$P(S, B|\boldsymbol{w}) = P(B|\boldsymbol{w})P(S|B, \boldsymbol{w}) \tag{4.3}$$

$$= P(B|\boldsymbol{w}) \prod_{\langle i,j \rangle} P(S_{\langle i,j \rangle}|B_{\langle i,j \rangle}) \tag{4.4}$$

$$= P(B|\boldsymbol{w}) \prod_{\langle i,j \rangle \notin T_B} \prod_{k:\delta_k=-1} F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \times \prod_{\langle i,j \rangle \in T_B} \prod_{k:\delta_k=+1} F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \tag{4.5}$$

$$= P(B|\boldsymbol{w}) \prod_{\langle i,j \rangle \notin T_B} \prod_{k:\delta_k=-1} F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \times \prod_{\langle i,j \rangle \in T_B} \prod_{k:\delta_k=-1} F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \times \prod_{\langle i,j \rangle \in T_B} \frac{\prod_{k:\delta_k=+1} F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k)}{\prod_{k:\delta_k=-1} F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k)} \tag{4.6}$$

$$= P(B|\boldsymbol{w}) \prod_{\langle i,j \rangle} \prod_{k:\delta_k=-1} F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \times \prod_{\langle i,j \rangle \in T_B} \prod_{k} F_k^{\delta_k}(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \tag{4.7}$$

$$= K(S|\boldsymbol{w}) \prod_{\langle i,j \rangle \in T_B} \prod_{k} F_k^{\delta_k}(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \tag{4.8}$$

The joint probability is factorized first by the chain rule (4.3), then over factors defined for each active span (4.4 and 4.5). In Equation (4.6), we introduce an additional term representing the product of distituent factors (for $k : \delta_k = -1$) over constituent spans (for $\langle i, j \rangle \in T_B$). We first multiply the additional term in the first part of the equation, then divide this term in the second part. Since span $\langle i, j \rangle$ either belongs to the tree span set T_B or not belongs to T_B, we can combine the two parts in Equation (4.6) to get the first term in Equation (4.7). Finally in Equation (4.8), we define the term $K(S|\boldsymbol{w})$ to

represent those products independent of B as

$$K(S|\boldsymbol{w}) = P(B|\boldsymbol{w}) \prod_{\langle i,j \rangle} \prod_{k:\delta_k=-1} F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \tag{4.9}$$

The rest products in Equation (4.8) are defined only over tree spans. In this way, we have reduced the complexity to evaluate the joint probability from all $O(n^2)$ spans to $O(n)$ tree spans for sentence with length n. The same trick can be found in the Appendix A.1 in (Klein, 2005).

As defined in Equation (4.1), factors are normalized locally over spans. One advantage of the locally normalized model is that the EM algorithm could be still used to learn the model parameters. The constant $K(S|\boldsymbol{w})$ in Equation (4.8) would be cancelled in the EM algorithm, which we will describe in the Section (4.1.2).

4.1.2 Parameter Estimation

In this section, we present the algorithm to estimate parameters for the feature-based CCM. Let $\mathcal{S}$ be the set of training sentences. As described in Section 2.2.1, we assign $P(B) = 0$ for $B \notin \mathcal{B}_T$. Under the maximum likelihood estimation, we want to find $\boldsymbol{w}$ to maximize the data log likelihood (ignoring non-tree-equivalent bracketings):

$$L(\mathcal{S}|\boldsymbol{w}) = \sum_{S \in \mathcal{S}} \log \sum_{B \in \mathcal{B}_T(S)} P(S, B|\boldsymbol{w}) \tag{4.10}$$

However, the summation of hidden variable B is inside the logarithm operator, resulting in the complicated expressions for the analytical solution. Instead, we use the Expectation-Maximization (EM) algorithm to solve the problem approximately.

Given current model parameters $\boldsymbol{w}^{old}$ in each iteration of EM, we seek new parameter $\boldsymbol{w}$ to maximize the expectation of the completed-data log likelihood:

$$Q(\boldsymbol{w}, \boldsymbol{w}^{old}) = \sum_{S \in \mathcal{S}} \sum_{B \in \mathcal{B}_T(S)} P(B|S, \boldsymbol{w}^{old}) \log P(S, B|\boldsymbol{w}) \tag{4.11}$$

E-Step

The E-step evaluates the posterior probability $P(B|S, \boldsymbol{w}^{old})$ given fixed $\boldsymbol{w}^{old}$. We can use the (modified) inside-outside algorithm (Lari and Young, 1990) to efficiently calculate the expected counts for each factor. To simplify derivations, we define

$$\phi_{\langle i,j\rangle} = \prod_k F_k^{\delta_k}(S_{\langle i,j\rangle}|\boldsymbol{w}_k^{old}) \tag{4.12}$$

For sentence S with length l, the inside probability IN can be calculated bottom-up recursively

$$\text{IN}_{\langle i,j\rangle} = \begin{cases} \phi_{\langle i,j\rangle}, & \text{if } j-i=1 \\ \sum_{k=i+1}^{j-1} \phi_{\langle i,j\rangle}\, \text{IN}_{\langle i,k\rangle}\, \text{IN}_{\langle k,j\rangle}, & \text{if } j-i>1 \end{cases}$$

The outside probability OUT can be calculated top-down recursively

$$\text{OUT}_{\langle i,j\rangle} = \begin{cases} 1, & \text{if } j-i=l \\ \sum_{k=0}^{i-1} \phi_{\langle k,j\rangle}\, \text{OUT}_{\langle k,j\rangle}\, \text{IN}_{\langle k,i\rangle} + \sum_{k=j+1}^{l} \phi_{\langle i,k\rangle}\, \text{OUT}_{\langle i,k\rangle}\, \text{IN}_{\langle j,k\rangle}, & \text{if } j-i<l \end{cases}$$

The fraction of trees that contain the span $\langle i,j\rangle$ as a constituent can be calculated as[1]:

$$r[\phi_{\langle i,j\rangle}] = \text{IN}_{\langle i,j\rangle} \times \text{OUT}_{\langle i,j\rangle} \,/\, \text{IN}_{\langle 0,l\rangle} \tag{4.13}$$

For each span $\langle i,j\rangle$, assuming the feature vector for factor $F_k(S_{\langle i,j\rangle})$ is $\boldsymbol{v}$, we accumulate the following expected counts for factor F_k:

$$e[F_k(\boldsymbol{v}), S_{\langle i,j\rangle}] = \begin{cases} r[\phi_{\langle i,j\rangle}], & \text{if } \delta_k = +1 \\ 1 - r[\phi_{\langle i,j\rangle}], & \text{if } \delta_k = -1 \end{cases} \tag{4.14}$$

We denote $e[F_k(\boldsymbol{v})]$ as the accumulated expected counts for factor F_k over training set.

We do not consider empty spans in the above calculation of inside/outside probabilities. Since the empty spans do not depend on tree structures, we just add expected count 1 for each distituent factor and 0 for each constituent factor over empty spans.

[1]There is a notation error in our previous paper (Huang et al., 2012), which is fixed here.

M-Step

The objective in M-step is to tune $\boldsymbol{w}$ to maximize the expected complicated-data log likelihood together with the regularization terms:

$$Q(\boldsymbol{w}, \boldsymbol{w}^{old}) - \lambda \|\boldsymbol{w}\|_1 \tag{4.15}$$

where λ is a non-negative coefficient for the ℓ_1-norm of $\boldsymbol{w}$. Because of the high-dimensional feature space, we use ℓ_1-norm of weight vector $\boldsymbol{w}$ as regularization terms to control the model complexity. The regularization terms can serve as automatic feature selector, leading to learn compact models. The ℓ_1-norm is preferred than the ℓ_2-norm since the former norm leads to much sparser model (Zou and Hastie, 2005).

In traditional CCM, model parameters (multinomial distribution probabilities) are estimated by normalizing relative frequencies in the M-step. In the feature-based model, we use gradient-based search algorithm to optimize the above objective function numerically. For differentiable objective functions, we may apply the Limited-memory BFGS (Nocedal, 1980) algorithm to optimize. Due to the ℓ_1 regularization term, however, the objective in Equation (4.15) is not differentiable at $\boldsymbol{w} = \mathbf{0}$. So we use the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) method (Andrew and Gao, 2007) to deal with ℓ_1-norm optimization. We use the open-source C++ implementation `libLBFGS`[2] in experiments.

The optimization process needs to calculate the gradient of $Q(\boldsymbol{w}, \boldsymbol{w}^{old})$ (without the regularization terms) respect to the weight vector $\boldsymbol{w}$. Mathematically, considering Equation (4.8) and (4.12), we have

$$\begin{aligned} P(B|S, \boldsymbol{w}^{old}) &= \frac{P(S, B|\boldsymbol{w}^{old})}{\sum_{B' \in \mathcal{T}_{\mathcal{B}}(\mathcal{S})} P(S, B'|\boldsymbol{w}^{old})} \\ &= \frac{K(S|\boldsymbol{w}^{old}) \prod_{\langle i,j \rangle \in T_B} \prod_k F_k^{\delta_k}(S_{\langle i,j \rangle}|\boldsymbol{w}_k^{old})}{\sum_{B' \in \mathcal{B}_T(S)} K(S|\boldsymbol{w}^{old}) \prod_{\langle i,j \rangle \in T_B} \prod_k F_k^{\delta_k}(S_{\langle i,j \rangle}|\boldsymbol{w}_k^{old})} \\ &= \frac{\prod_{\langle i,j \rangle \in T_B} \phi_{\langle i,j \rangle}}{\sum_{B' \in \mathcal{B}_T(S)} \prod_{\langle i,j \rangle \in T_B} \phi_{\langle i,j \rangle}} \end{aligned} \tag{4.16}$$

[2] http://www.chokkan.org/software/liblbfgs/

in which the constant $K(S|\boldsymbol{w}^{old})$ could be cancelled in derivation. We can substitute Equation (4.9) into the definition of Q (Equation (4.11)) and expand Q as

$$\begin{aligned}
Q(\boldsymbol{w}, \boldsymbol{w}^{old}) &= \sum_{S} \sum_{B \in \mathcal{B}_T(S)} P(B|S, \boldsymbol{w}^{old}) \log P(S, B|\boldsymbol{w}) \\
&= \sum_{S} \sum_{B \in \mathcal{B}_T(S)} P(B|S, \boldsymbol{w}^{old}) \log \left[K(S|\boldsymbol{w}) \prod_{\langle i,j \rangle \in T_B} \prod_{k} F_k^{\delta_k}(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \right] \\
&= \sum_{S} \sum_{B \in \mathcal{B}_T(S)} P(B|S, \boldsymbol{w}^{old}) \Bigg[\log P(B|\boldsymbol{w}) + \sum_{\langle i,j \rangle} \sum_{k:\delta_k=-1} \log F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \\
&\qquad + \sum_{\langle i,j \rangle \in T_B} \sum_{k} \delta_k \log F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \Bigg] \\
&= C + \sum_{k:\delta_k=+1} Q_k^{(c)}(\boldsymbol{w}, \boldsymbol{w}^{old}) + \sum_{k:\delta_k=-1} Q_k^{(d)}(\boldsymbol{w}, \boldsymbol{w}^{old}) \qquad (4.17)
\end{aligned}$$

where C is a constant value independent of $\boldsymbol{w}$, and $Q_k^{(c)}$ and $Q_k^{(d)}$ represent the corresponding quantities for the constituent and distituent factors, which are defined as:

$$\begin{aligned}
Q_k^{(c)}(\boldsymbol{w}, \boldsymbol{w}^{old}) &= \sum_{S} \sum_{B \in \mathcal{B}_T(S)} P(B|S, \boldsymbol{w}^{old}) \sum_{\langle i,j \rangle \in T_B} \log F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \qquad (4.18) \\
&= \sum_{S} \sum_{B \in \mathcal{B}_T(S)} \sum_{\langle i,j \rangle \in T_B} r[\phi_{\langle i,j \rangle}] \log F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \qquad (4.19) \\
&= \sum_{\boldsymbol{v} \in \mathcal{V}_k} e[F_k(\boldsymbol{v})] \log F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \qquad (4.20) \\
Q_k^{(d)}(\boldsymbol{w}, \boldsymbol{w}^{old}) &= \sum_{S} \sum_{B \in \mathcal{B}_T(S)} P(B|S, \boldsymbol{w}^{old}) \Bigg[\sum_{\langle i,j \rangle} \log F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \\
&\qquad - \sum_{\langle i,j \rangle \in T_B} \log F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \Bigg] \qquad (4.21) \\
&= \sum_{S} \sum_{B \in \mathcal{B}_T(S)} \sum_{\langle i,j \rangle \in T_B} (1 - r[\phi_{\langle i,j \rangle}]) \log F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \qquad (4.22) \\
&= \sum_{\boldsymbol{v} \in \mathcal{V}_k} e[F_k(\boldsymbol{v})] \log F_k(S_{\langle i,j \rangle}|\boldsymbol{w}_k) \qquad (4.23)
\end{aligned}$$

in which the set $\mathcal{V}_k$ contains all appeared values in the training set of the k^{th} factor F_k, and $e[F_k(\boldsymbol{v})]$ is the accumulated expected counts for factor F_k.

Therefore, the expansion of Q in Equation (4.17) can be written as

$$Q(\boldsymbol{w}, \boldsymbol{w}^{old}) = C + \sum_{k} Q_k(\boldsymbol{w}, \boldsymbol{w}^{old}) \tag{4.24}$$

$$\text{where} \quad Q_k(\boldsymbol{w}, \boldsymbol{w}^{old}) = \sum_{\boldsymbol{v} \in \mathcal{V}_k} e[F_k(\boldsymbol{v})] \log F_k(S_{\langle i,j \rangle} | \boldsymbol{w}_k) \tag{4.25}$$

in which Q_k is the corresponding component for the k^{th} factor. The probabilities of factors are multiplied together, so the logarithm term in the above equation can be decomposed into the sum of the logarithm of each factor probability. Furthermore, the ℓ_1-norm term in Equation (4.15) can be also written as the sum of ℓ_1-norm of the corresponding weights for each factor. As a result, optimizing the overall objective function is equivalent to optimize objective functions for each factor. This does not only allow us to simplify the derivations and computation of the gradient, but also makes it possible to use different regularization parameter λ_k for different factors F_k. Since different factors have different feature numbers and feature spaces, individual regularization may improve the overall performance.

Finally, the gradient of Q_k respect to the corresponding feature weight vector $\boldsymbol{w}_k$ for factor F_k can be computed as follows:

$$\boldsymbol{\nabla}_{\boldsymbol{w}_k}(Q_k) = \sum_{\boldsymbol{v} \in \mathcal{V}_k} e[F_k(\boldsymbol{v})] \times \boldsymbol{\Delta}_{\boldsymbol{v}}(\boldsymbol{w}_k) \tag{4.26}$$

$$\boldsymbol{\Delta}_{\boldsymbol{v}}(\boldsymbol{w}_k) = \boldsymbol{f}_k(\boldsymbol{v}) - \sum_{\boldsymbol{v}' \in \mathcal{V}_k} F_k(\boldsymbol{v}') \boldsymbol{f}_k(\boldsymbol{v}') \tag{4.27}$$

where $e[F_k(\boldsymbol{v})]$ is the expected counts accumulated in the E-step. The similar derivation can be found in (Berg-Kirkpatrick et al., 2010).

Rich features can be easily incorporated in this feature-based model. In next section, we show the feature templates used our experiments.

4.2 Feature Templates

4.2.1 Basic features

There are two kinds of features: constituent features, with prefix {c:}; and distituent features, with prefix {d:}. Features in the two categories are active only if the span enclose constituent or distituent respectively. The basic feature templates are listed as follows with their names and descriptions. A running example, span $\langle 1, 3\rangle$ in "$_0$RB$_1$DT$_2$NN$_3$", is also shown for each feature template.

• const: This constant feature always takes value 1 for any given span. We use this feature to measure the number of spans.

• seq[n]: This indicating feature is active for sequence enclosed by span with size n. If $n = 0$, then sequences with any length are considered.

seq2	...	DT_JJ	DT_NN	RB_DT	...
value	...	0	1	0	...

• lx[n]/rx[n]: The indicating feature for the preceding/following n terminals (left/right context), where $\diamond$ represents sentence boundary.

lx2	...	◇_◇	◇_RB	RB_DT	...
value	...	0	1	0	...

rx2	...	DT_NN	NN_◇	◇_◇	...
value	...	0	0	1	...

• lb[n]/rb[n]: The left/right n boundary terminals inside given span. If the length of span is less than n, then this feature template is not activated.

lb2	...	RB_DT	DT_NN	TO_VB	...
value	...	0	1	0	...

rb1	...	RB	DT	NN	...
value	...	0	0	1	...

4.2.2 Composite features

Basic features can be composited to more complicated features. We define two composition operators: join (.), and concatenation (+). For the join operator, the composited feature space is the Cartesian product of the feature spaces of the two operands. For the concatenation operator, the composited feature space is the concatenation of the operands' feature spaces.

Here we use an example to demonstrate the difference between join operator and concatenation operator. Assume there are 3 possible values ($\diamond$, RB, DT) for feature lx1, and 3 possible values (DT, NN, $\diamond$) for feature rx1. We consider feature vectors of the two operators for span $\langle 1, 3\rangle$ in "$_0$RB$_1$DT$_2$NN$_3$". The joined feature space has $3 \times 3 = 9$ dimensions:

lx1.rx1	$\diamond$.{DT,NN,$\diamond$}			RB.{DT,NN,$\diamond$}			DT.{DT,NN,$\diamond$}		
value	0	0	0	0	0	1	0	0	0

The concatenated feature space has $3 + 3 = 6$ dimensions:

lx1+rx1	$\diamond$	RB	DT	DT	NN	$\diamond$
value	0	1	0	0	0	1

We only allow compositions with join operators followed by concatenation operators. In this representation, the original CCM could be represented as: {c:seq0, d:seq0, c:lx1.rx1, d:lx1.rx1}. We show templates used in experiments in next subsection.

4.2.3 Templates in Experiments

Various knowledge can be incorporated into the feature-based model. However, since there are huge feature combinations, we can not enumerate them in experiments. In experiments, we use a restricted set of features described as follows.

The first feature set used in experiments is the sequence with length up to 5: {`seq1`, `seq2`, `seq3`, `seq4`, `seq5`}. Note that the original CCM consider sequences with arbitrary lengths, while we restrict the maximal sequence length to be 5. Since most of the longer sequences occurs only once or twice in the training set, we discard them to reduce the memory usage and disk spaces. For long sentences, we find the following boundary features, which appear much more frequently than sequence features, play important role in experiments (see subsection 4.3.5).

Boundary words have been proven useful for detecting phrase boundaries in supervised setting (Xiong et al., 2010; He et al., 2010). We introduce this idea to unsupervised grammar induction. The features used in experiments are combinations of left boundary and right boundary words with lengths up to 2: {`lb1`, `lb2`, `rb1`, `rb2`, `lb1.rb1`, `lb1.rb2`, `lb2.rb1`, `lb2.rb2`}.

The original CCM also consider the pair of preceding one word and following one word as contexts. We consider combinations of left context and right context words with lengths up to 2: {`lx1`, `lx2`, `rx1`, `rx2`, `lx1.rx1`, `lx1.rx2`, `lx2.rx1`, `lx2.rx2`}. The special sentence boundary token $\diamond$ is introduced when needed.

The last feature used is the constant feature {`const`}. The constant feature always takes value 1 for each span.

Overall, we define 2 constituent and 2 distituent factors in the feature-based model. The first constituent (distituent) factor is the concatenation of the sequence features, the boundary features, and the constant feature: {`seq1+...+seq5+lb1+...+lb2.rb2+const`}. These two factors are denoted as $F_{\texttt{c:s}}$ and $F_{\texttt{d:s}}$ respectively. The second constituent (distituent) factor is the concatenation of the context features and the constant feature: {`lx1+...+lx2.rx2+const`}. These two factors are represented as $F_{\texttt{c:x}}$ and $F_{\texttt{d:x}}$ respectively.

4.3 Experiments

4.3.1 Datasets and Settings

We carry out experiments on the Wall Street Journal portion of the Penn English Treebank (Marcus et al., 1993), in which sections 02-21 are used as the training set, section 00 is used as the development set, and section 23 is used as the test set. We remove null elements (such as "*-1" and "-NONE-") in treebank, since they are only for linguistic purposes and not readable for human learner. In addition, we remove words acting as punctuations in sentences if the part-of-speech (POS) tag is one of the follows:

, . : " " $ # -LRB- -RRB-

where the last two POS tags represent the left brackets and right brackets respectively. We follow previous practices (Klein and Manning, 2002; Klein, 2005) and remove punctuations for simplicity. Finally, tree nodes dominating no elements are pruned. The detailed preprocessing step could be seen in (Klein, 2005). For comparison, we build various datasets with sentences lengths no more than 10, 20, 30, 40 words after removing null elements and punctuations. Table 4.1 gives the statistics for each dataset. Figure 4.1 shows an example of the parse tree found in the training set.

Dataset	Train		Dev		Test	
	# sent	# word	# sent	# word	# sent	# word
PTB10	5899	41701	265	1875	398	2649
PTB20	20243	266785	992	13309	1286	16591
PTB30	32712	579241	1573	27929	2028	35148
PTB40	37561	746844	1809	35999	2338	45813

Table 4.1: Penn treebank data statistics.

The baseline system is the original EM-based constituent-context model (Klein and Manning, 2002; Klein, 2005). EM algorithm is sensitive to the initial condition, so we adopt the same uniform-split initialization. Following previous work, we use the part-of-

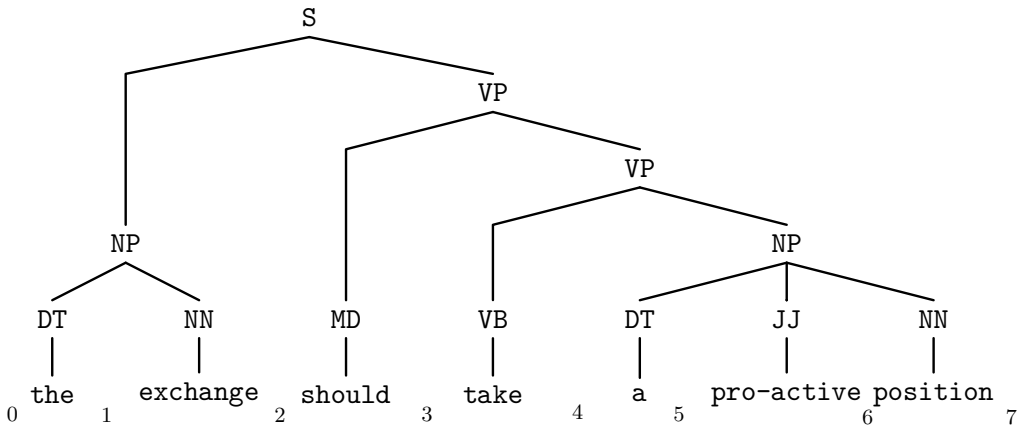


Figure 4.1: An example of reference tree.

speech tag sequences instead of raw words as the input of the baseline system and our induction system. We also report performance of other trivial baselines for comparison, including the left-branching baseline and the right-branching baseline. Figure 4.2 and 4.3 show the corresponding left-branching tree and right-branching tree of the above reference tree (Figure 4.1), where the special nonterminal `Con` represents the constituent placeholder in the tree node. For English, right-branching happens to be a strong baseline (e.g. we can see from the figures that the right-branching tree has similar structures to the reference tree). However, other languages may have other branching biases (Klein, 2005). We also evaluate the performance of the binarized treebank, as the upper bound of any binary-tree induction system. Figure 4.4 shows the binarized tree of the mentioned reference tree (shown in Figure 4.1), in which the new introduced nonterminal `NP-DT` binarized the original flat span $\langle 4, 7 \rangle$ into small ones.

For both the baseline CCM and proposed feature-based CCM, we tune smoothing values on the development set for constituent factors from $\{2, 8, 20\}$, and those for distituent factors from $\{8, 20, 40, 80, 160\}$. There are many parameter combinations, so we first fix the distituent smoothing value to be 80 and tune constituent smoothing values, then tune distituent smoothing values with the tuned constituent smoothing value. The results reported in this thesis are the best tuned ones.

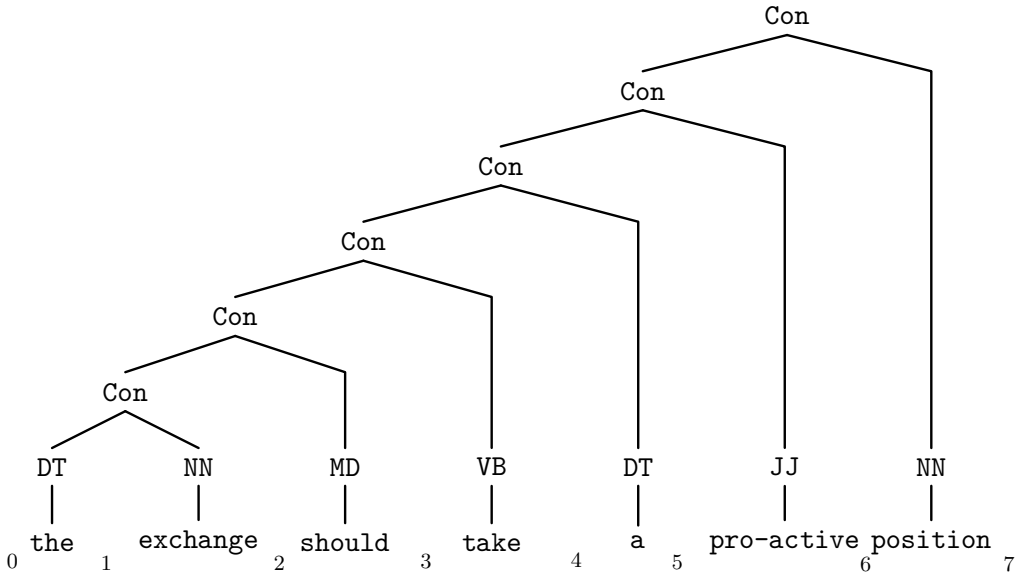


Figure 4.2: An example of left branching tree.

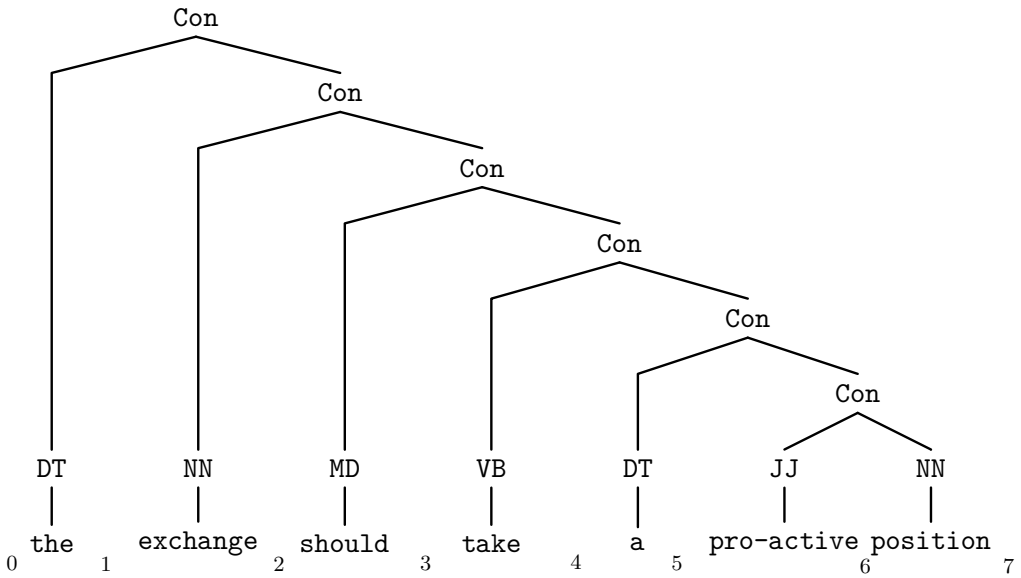


Figure 4.3: An example of right branching tree.

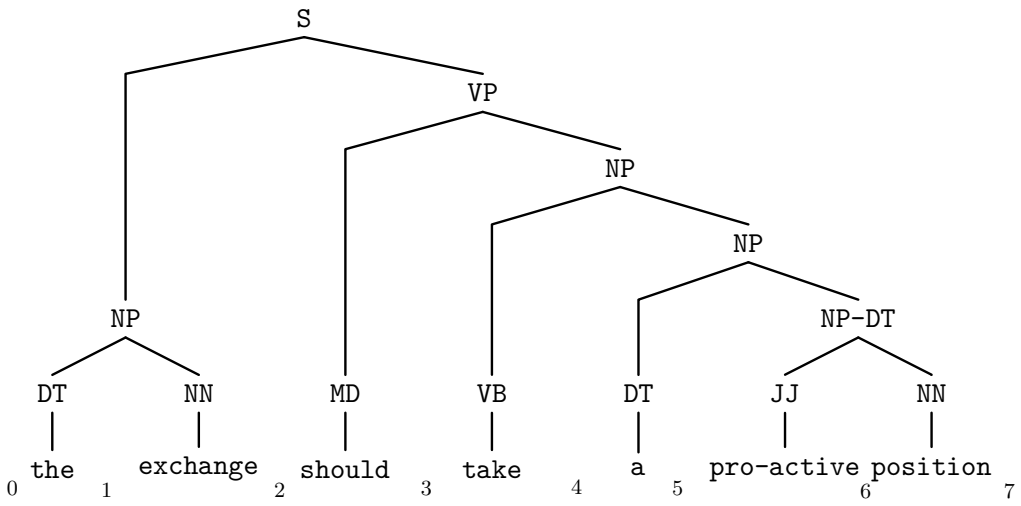


Figure 4.4: An example of binarized reference tree, the upper bound of any binary induction system.

For feature-based model (F-CCM), we still use uniform-split strategy to initialize probabilities in the first E-step, and set all weights to zero as the initial point of the gradient-based search algorithm in the M-step. As the standard machine learning pipeline, for both baseline models and the proposed models, we perform learning on the training set, select the model with the best performance on the development set, and report the final result of selected model on the test set. After training the feature-based models on the training set, we obtain the weights for each feature dimensions. We then use these weights to parse and induce trees on the development and test datasets.

For the four factors used in feature-based CCM, we select regularization parameters λ from set {0.03, 0.1, 0.3, 1, 3, 10, 30}[3]. The use of development for tuning is a reasonable way for selecting model parameters. We choose the parameters that achieve the highest development score as final regularization values and report the corresponding evaluation metrics on the test datasets.

[3]In our previous work (Huang et al., 2012), we did not regularize factors $F_{\texttt{c:x}}$ and $F_{\texttt{d:x}}$. In this thesis, we perform regularization for these two factors as well and rerun some experiments, so some of the results in this thesis are different from our previous reported ones.

4.3.2 Evaluation Metrics

The evaluation objective is sometimes unclear for unsupervised grammar learning tasks, which depends the following processing tasks. Moreover, the objective function that unsupervised models try to optimize may differ from the evaluation metrics (Liang and Klein, 2008). We follow previous unsupervised constituency tree induction approaches (Klein, 2005; Smith and Eisner, 2004; Golland et al., 2012) and evaluate the induced trees from our system against the annotated treebank. Since our models only induce the set of bracketings for raw strings without annotated labels, we report the unlabeled precision (P), unlabeled recall (R), and their harmonic mean (F_1). These metrics differ from the standard PARSEVAL metric (Black et al., 1991) in following ways: constituent spans contain single words are discarded and multiplicity of brackets is ignored in evaluation.

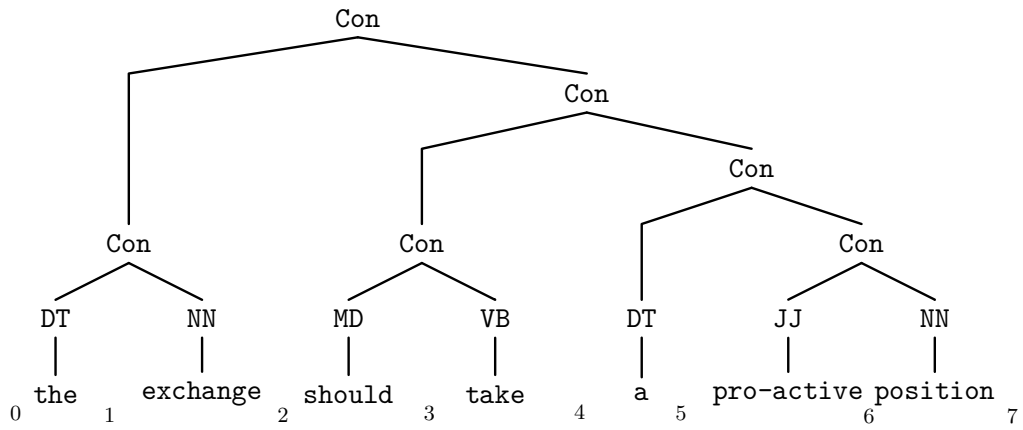


Figure 4.5: An example of candidate tree.

In detail, we represent a parse tree T to be a set of unlabeled constituent brackets. Each tree node corresponds to one span $\langle i, j \rangle$ over the constituent that the node covers. Terminal (word) and preterminal (POS tag) nodes are excluded, as are nonterminal nodes which dominate only a single terminal. Let $\mathcal{G} = \{\mathcal{G}_i\}$ and $\mathcal{C} = \{\mathcal{C}_i\}$ denote the set of span representations for the golden tree bank and the system output respectively, and

$\mathcal{M} = \{\mathcal{M}_i\} = \{\mathcal{G}_i \cap \mathcal{C}_i\}$ denote the matched span set, then the unlabeled precision, recall and F_1 could be calculated as follows

$$P = \frac{\sum_i |\mathcal{M}_i|}{\sum_i |\mathcal{C}_i|} = \frac{\sum_i |\mathcal{C}_i \cap \mathcal{G}_i|}{\sum_i |\mathcal{C}_i|} \tag{4.28}$$

$$R = \frac{\sum_i |\mathcal{M}_i|}{\sum_i |\mathcal{G}_i|} = \frac{\sum_i |\mathcal{C}_i \cap \mathcal{G}_i|}{\sum_i |\mathcal{G}_i|} \tag{4.29}$$

$$F_1 = \frac{2\,P\,R}{P + R} \tag{4.30}$$

Note that the above $P/R/F_1$ are calculated over all sentences in the tree bank.

We use examples to show how to evaluate these metrics. The reference tree in Figure 4.1 and the candidate tree in Figure 4.5 can be represented as following span sets

Constituent	Ref	Cand	Matched
`DT NN`	$\langle 0, 2\rangle$	$\langle 0, 2\rangle$	$\langle 0, 2\rangle$
`MD VB`	-	$\langle 2, 4\rangle$	-
`JJ NN`	-	$\langle 5, 7\rangle$	-
`DT JJ NN`	$\langle 4, 7\rangle$	$\langle 4, 7\rangle$	$\langle 4, 7\rangle$
`VB DT JJ NN`	$\langle 3, 7\rangle$	-	-
`MD VB DT JJ NN`	$\langle 2, 7\rangle$	$\langle 2, 7\rangle$	$\langle 2, 7\rangle$
`DT NN MD VB DT JJ NN`	$\langle 0, 7\rangle$	$\langle 0, 7\rangle$	$\langle 0, 7\rangle$
Total	$\lvert\mathcal{G}\rvert = 5$	$\lvert\mathcal{C}\rvert = 6$	$\lvert\mathcal{M}\rvert = 4$

As a result, the precision and recall for this example are $P = \frac{4}{6}$ and $R = \frac{4}{5}$ respectively. In the similar way, the number of matched spans, total spans and the precision and recall are (1) $|\mathcal{M}_l| = 2$, $|\mathcal{C}_l| = 6$, $P_l = \frac{2}{6}$, $R_l = \frac{2}{5}$, for the left-branching tree (Figure 4.2); (2) $|\mathcal{M}_r| = 4$, $|\mathcal{C}_r| = 6$, $P_r = \frac{4}{6}$, $R_r = \frac{4}{5}$, for the right-branching tree (Figure 4.3); (3) $|\mathcal{M}_u| = 5$, $|\mathcal{C}_u| = 6$, $P_u = \frac{5}{6}$, $R_u = \frac{5}{5}$, for the binarized reference tree (Figure 4.4). From this example, we can see that the left-branching tree has bad P/R. In contrast, the right-branching baseline matches the reference tree well (as good as the induced tree in this example). For any binarized reference tree, the recall is always 100% since it never misses any span in the reference.

4.3.3 Induction Results

Table 4.2 shows the experimental results on the Penn English Treebank datasets of different length limits (PTB10, PTB20, PTB30, PTB40). LBranch and RBranch rows show the left branching and right branching baselines. UBound rows show the results of binarized treebank, which is the upper bound of any grammar induction systems that output binary-trees. We reimplement the baseline CCM, which achieves comparable performance compared to previous reported results (Klein, 2005). The results of feature-based CCM are presented in the F-CCM rows.

From these results, we observe that the left branching baseline are bad for English language, while the right branching baseline achieves relative good performance for various datasets. The upper bound F_1 metrics range from about 85% to 89%, which is lower than expected. The annotation guild line of Penn Treebank forces trees to be relative flat trees with big phrase structure (usually the noun phrases). The binarized treebank gets low precisions for these cases, especially for long sentences.

The original CCM performs much better than the right branching baseline on short sentences while the performance decreases dramatically on longer sentences (even worse than than the right branching baseline). These evidences show that the single multinomial distributions for constituents and distituents are not able to capture complicated tree structures appeared in long sentences. In contrast, our proposed F-CCM achieves much better performance than the CCM on long sentences. The precision, recall and F_1 metrics of F-CCM all outperform CCM and the right branching baseline in a large gap on large datasets. These results demonstrate the effectiveness of the feature-based models.

The performance of F-CCM is slightly worse than CCM on PTB10. The reason might be that we use shorter sequences (maximal 5). We have carried out experiments of F-CCM with exactly the same features as CCM, the performance of F-CCM is almost the same as CCM on all datasets (less than $\%3$ F_1 differences). The feature templates used to report the final results in Table 4.2 are those described in subsection 4.2.3.

Dataset	Train			Dev			Test		
PTB10	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$
LBranch	25.60	32.46	28.62	25.65	32.42	28.64	27.01	35.23	30.58
RBranch	55.08	69.83	61.58	56.96	71.98	63.59	53.89	70.28	61.00
UBound	78.88	100.0	88.20	79.13	100.0	88.35	76.68	100.0	86.80
CCM	64.85	82.21	**72.50**	65.90	83.28	**73.58**	62.11	81.00	**70.30**
F-CCM	64.32	81.53	71.91	65.53	82.81	73.16	61.66	80.42	69.80
PTB20	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$
LBranch	15.16	19.95	17.22	15.33	20.21	17.43	15.13	19.97	17.21
RBranch	42.57	56.04	48.39	42.07	55.47	47.85	42.14	55.64	47.96
UBound	75.97	100.0	86.35	75.85	100.0	86.26	75.74	100.0	86.20
CCM	43.08	56.71	48.96	42.61	56.18	48.46	42.25	55.78	48.08
F-CCM	52.67	69.33	**59.86**	52.63	69.39	**59.86**	51.93	68.56	**59.10**
PTB30	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$
LBranch	11.70	15.60	13.37	11.94	15.82	13.61	11.68	15.52	13.33
RBranch	37.37	49.82	42.70	37.51	49.71	42.76	37.30	49.57	42.57
UBound	75.01	100.0	85.72	75.47	100.0	86.02	75.25	100.0	85.88
CCM	37.63	50.17	43.01	37.96	50.30	43.27	37.32	49.59	42.59
F-CCM	42.77	57.01	**48.87**	42.83	56.75	**48.82**	42.20	56.07	**48.15**
PTB40	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$
LBranch	10.56	14.12	12.08	10.78	14.35	12.31	10.45	13.95	11.95
RBranch	35.46	47.45	40.59	35.49	47.26	40.54	35.63	47.54	40.73
UBound	74.74	100.0	85.54	75.09	100.0	85.77	74.96	100.0	85.69
CCM	29.22	39.10	33.44	29.43	39.19	33.62	28.95	38.62	33.10
F-CCM	39.70	53.12	**45.44**	39.80	53.00	**45.46**	39.46	52.64	**45.10**

Table 4.2: Induction results of feature-based CCM.
`[Data]` PTB10, PTB20, PTB30, PTB40.
`[Rows]` LBranch: left branching tree; RBranch: right branching tree; UBound: binarized treebank, which is the upper bound of any grammar induction systems that output binary trees; CCM: the original constituent-context model; F-CCM: the proposed feature-based CCM.
`[Columns]` P: overall precision; R: overall recall; F_1: overall F-score.

4.3.4 Grammar sparsity

The regularization terms can serve as the feature selector. In this section, we compare the sparsity of learned grammars between various regularization coefficients on PTB10. As mentioned in Section 4.1.2, we use different regularization coefficients for different factors. Since there are too many results of parameter combinations to show here, we only show the results that all factors use the same regularization coefficients in Table 4.3. The dimension of the sequence factors ($F_{\texttt{c:s}}$ and $F_{\texttt{d:s}}$) is 72289, and the dimension of the context factors ($F_{\texttt{c:x}}$ and $F_{\texttt{d:x}}$) is 54439. We report the number of weights with non-zero values as the measurement of grammar sparsity.

λ	$F_{\texttt{c:s}}$	$F_{\texttt{d:s}}$	$F_{\texttt{c:x}}$	$F_{\texttt{d:x}}$	Dev F_1	Test F_1
0.03	68963	71622	52806	54199	72.33	68.19
0.1	57907	69683	47240	52672	73.16	69.80
0.3	34954	57316	32120	46828	72.40	68.85
1	11738	27735	13713	24113	72.82	69.70
3	4125	10064	5621	10228	72.75	70.15
10	1498	3325	2345	4323	70.46	67.44
30	630	1231	1002	1874	67.20	62.36

Table 4.3: Sparsity of the induced grammars. The λ column gives the regularization coefficients, the middle four columns show the number of non-zero weights of each factor, and the last two columns show the corresponding F_1 value on the development set and test set respectively.

From this table, we can see that the ℓ_1-norm with larger λ leads to sparser model with less non-zero dimensions. However, if the non-zero weights are penalized too heavy, the feature-based model would underfit the training data and ends with bad development F_1. The suitable value of regularization coefficient ($\lambda = 0.1$) can be selected by the development set. Another interesting observation is that the number of non-zero distituent factors ($F_{\texttt{d:s}}$ and $F_{\texttt{d:x}}$) is much greater than constituent factors ($F_{\texttt{c:s}}$ and$F_{\texttt{c:x}}$). For tree with yield length n, there are $O(n)$ constituent spans and $O(n^2)$ distituent spans, so the feature-based model needs more distituents to encode the probability distributions.

4.3.5 Feature Analysis

In subsection 4.2, we designed (1) constant feature, (2) boundary features, (3) sequence features, and (4) context features for the feature-based CCM. We examine which kind of features works well for F-CCM in this subsection. We subtract each feature set from the final feature set and rerun the experiments. The experimental results are shown in Table 4.4. The CCM and F-CCM results are also given for comparison.

If the constant feature is excluded from the feature set, the performances slightly decrease on all datasets. We have checked the weight of the constant feature and found that the weight is quite small (less than 10^{-6}), so this feature does not show much discriminating ability.

The boundary features affect F-CCM very much, especially for short sentences. For short spans, usually the boundary words can determine the phrase category, such as the noun phrases usually begin with articles and end with nouns. For long sentences, the boundary features still has significant impact, so we conclude the boundary words could help for unsupervised grammar induction. Note that we use more complex context features than CCM, so the performances without boundary features are still better than the original CCM on long sentences.

One interesting observation is that excluding the sequence features does not hurt performance much, and even slightly improve the performance on long sentences. Since we design boundary features to capture constituent contents, the sequence features may be duplicated. In addition, as long sequences occur a few times in the training corpus, the parameter estimation may be unreliable.

The context features play the most important role in feature-based CCM, since the performances drops most if the context features are excluded from the feature set. This gives evidence to the claim that constituents appear in constituent contexts, which is the motivation of distributional clustering.

In summary, the boundary and context features are the most important features.

Dataset	Train			Dev			Test		
PTB10	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$
CCM	64.85	82.21	72.50	65.90	83.28	73.58	62.11	81.00	70.30
F-CCM	64.32	81.53	71.91	65.53	82.81	73.16	61.66	80.42	69.80
-const	63.82	80.90	71.35	65.16	82.34	72.75	60.95	79.49	69.00
-bdr	50.18	63.61	56.10	50.31	63.58	56.17	48.96	63.85	55.42
-seq	63.39	80.36	70.88	65.28	82.50	72.88	61.35	80.01	69.45
-ctx	42.43	53.79	47.44	41.86	52.90	46.74	41.94	54.69	47.47
PTB20	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$
CCM	43.08	56.71	48.96	42.61	56.18	48.46	42.25	55.78	48.08
F-CCM	52.67	69.33	59.86	52.63	69.39	59.86	51.93	68.56	59.10
-const	52.49	69.09	59.65	52.41	69.10	59.61	51.62	68.15	58.74
-bdr	41.15	54.17	46.77	41.43	54.62	47.12	40.82	53.90	46.46
-seq	51.56	67.86	58.60	51.49	67.89	58.56	50.76	67.02	57.77
-ctx	36.55	48.11	41.54	36.01	47.47	40.95	36.16	47.75	41.16
PTB30	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$
CCM	37.63	50.17	43.01	37.96	50.30	43.27	37.32	49.59	42.59
F-CCM	42.77	57.01	48.87	42.83	56.75	48.82	42.20	56.07	48.15
-const	42.49	56.64	48.55	42.64	56.50	48.60	41.86	55.63	47.77
-bdr	37.87	50.49	43.28	38.36	50.83	43.73	37.66	50.05	42.98
-seq	43.38	57.83	49.57	43.74	57.96	49.86	42.91	57.03	48.97
-ctx	32.06	42.73	36.63	31.95	42.33	36.41	32.77	43.56	37.40
PTB40	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$	$P(\%)$	$R(\%)$	$F_1(\%)$
CCM	29.22	39.10	33.44	29.43	39.19	33.62	28.95	38.62	33.10
F-CCM	39.70	53.12	45.44	39.80	53.00	45.46	39.46	52.64	45.10
-const	39.62	53.01	45.35	40.01	53.28	45.70	39.22	52.33	44.84
-bdr	37.06	49.58	42.41	37.55	50.00	42.89	36.83	49.13	42.10
-seq	40.82	54.62	46.72	41.18	54.84	47.04	40.31	53.78	46.08
-ctx	30.73	41.11	35.17	30.92	41.18	35.32	30.86	41.17	35.28

Table 4.4: Induction results of feature-based CCM for feature subtraction experiments.
`[Data]` PTB10, PTB20, PTB30, PTB40.
`[Rows]` CCM: the original constituent-context model; F-CCM: the proposed feature-based CCM; -const: all feature sets except constant feature; -bdr: all feature sets except boundary features; -seq: all feature sets except sequence features; -ctx: all feature sets except context features.
`[Columns]` P: overall precision; R: overall recall; F_1: overall F-score.

4.3.6 Discussion

Experiments show that we achieve better performance with traditional CCM while using much compact grammars. There are some issues we want to discuss here.

1. There are too many feature templates to explore, and we only test a few of them. Other kinds of features may improve the induction performance, such as words and stems (Headden III et al., 2009), and punctuations (Spitkovsky et al., 2011b; Ponvert et al., 2011). They can be easily added as features, although we have not tested them. In addition, we can also design manually rules as features to precisely control the induced tree and may further improve performances for particular annotation guild lines.

2. In previous approaches for unsupervised constituency grammar induction (Klein, 2005; Smith and Eisner, 2004; Golland et al., 2012), they tune parameters and evaluate metrics on the same dataset, which is problematic. As a result, we advocate using a separated development set to perform model selection.

3. The EM algorithm only find the sub-optima in the parameter space. Online EM algorithms have shown improvements over full EM on some unsupervised learning tasks (Liang and Klein, 2009; Spitkovsky et al., 2010a; Spitkovsky et al., 2010b). These ideas can be easily incorporated into our feature-based EM, with minor modification of the expected count calculation in the E-step. Another learning algorithm is the *Lateen EM* (Spitkovsky et al., 2011a), in which multiple objective functions are alternative optimized. We may simulate multiple objective functions using different regularization coefficients and alternatively optimize them.

4. ℓ_1-norm regularization is used to learn sparse and compact model. Bayesian learning methods are alternatively frameworks to learn compact grammars, which can be also applied for CCM inference.

4.4 Summary

In this chapter, we presented a feature-based model for CCM, in which various knowledge can be integrated as features. The local normalization nature makes it suitable to fit in the EM algorithm. The use of ℓ_1-norm regularization leads to compact grammars. We also proposed a reasonable model selection and evaluation framework. Experimental results demonstrated the proposed model achieved better performance compared to the CCM baseline especially on long sentences.

Chapter 5

Improved Combinatory Categorial Grammar Induction

Combinatory Categorial Grammar (CCG) is an expressive lexicalized grammar formalism which is able to capture long-range dependencies. Bisk and Hockenmaier (2012b) propose a simple robust CCG induction method, in which lexicons for each part-of-speech tags are generated first, then the Expectation-Maximization (EM) is used to estimate model parameters. They compare the full EM, the Viterbi EM and the k-best EM schemes and find that the k-best EM algorithm performs best.

In this chapter, we focus on the above approach and propose extensions and improvements. Specifically, we introduce boundary part-of-speech (POS) tags into the baseline model to capture lexical information of language. The boundary model and the basic model are combined together. We also perform nonparametric Bayesian inference based on the Pitman-Yor process to learn compact grammars. Experimental results demonstrate that the boundary models consistently improve the baseline models for all learning algorithms (full EM, k-best EM, and Bayesian inference). The Bayesian inference outperforms the full EM, but underperforms the k-best EM.

5.1 Grammar Generation

Bisk and Hockenmaier (2012b) propose a simple iterative lexicon generation algorithm from the golden part-of-speech (POS) tags. Due to the simplicity and effectiveness of this method, we also adopt it to generate lexicons in our method. We rephrase their algorithm with minor modifications in this section.

Only two atomic categories, `N` (nouns or noun phrases)[1], and `S` (sentences) are allowed in grammar. Conjunction words are expanded from a special conjunction category `conj`. Trees are all generated from a special start symbol `TOP`. In assumption, all strings are either nouns or sentences, i.e. they are generated from one of the two unary rules:

$$\texttt{TOP} \rightarrow \texttt{N} \qquad\qquad \texttt{TOP} \rightarrow \texttt{S}$$

In addition, we restrict that: (1) strings containing at least one verb must be parsed with the `TOP-S` rule; and (2) strings without any verb must be parsed with the `TOP-N` rule[2].

The initial CCG lexicon $\mathcal{L}^{(0)}$ is created manually by assigning atomic category `N` to nouns, `S` to verbs, and `conj` to conjunctions for fixed POS tags. The following is an example of initial lexicon for the English Penn Treebank tag set (Marcus et al., 1993):

$$\begin{aligned}
\texttt{N} &: \{\texttt{DT}, \texttt{NN}, \texttt{NNS}, \texttt{NNP}, \texttt{NNPS}, \texttt{PRP}\} \\
\texttt{S} &: \{\texttt{MD}, \texttt{VB}, \texttt{VBD}, \texttt{VBG}, \texttt{VBN}, \texttt{VBP}, \texttt{VBZ}\} \\
\texttt{conj} &: \{\texttt{CC}\}
\end{aligned}$$

Note that the tag `NNPS` (representing plural proper noun) and the tag `VBP` (representing verb of non-3rd person singular and in present tense) are missing in (Bisk and Hockenmaier, 2012b) but they are included in the treebank tag set.

The lexicon for atomic categories remains fixed after the initial lexicon $\mathcal{L}^{(0)}$ has been created. However, the categories that POS tags may acquire are updated iteratively during

[1] In formal English grammars, `NP` is often used to represent noun phrases(Hockenmaier and Steedman, 2007). Following (Bisk and Hockenmaier, 2012b), we do not distinguish noun phrase from nouns for efficiency. This simple treatment causes some problems, e.g. the determiners would be treated as adjuncts and then regarded optional, but actually they are needed for singular count nouns.

[2] Bisk and Hockenmaier (2012b) only make the first restriction.

induction. In each step, we create new category candidates for adjacent words, including: (a) modifiers, in the form of X|X; and (b) functors, in the form of X|Y for different categories X and Y. The motivation of modifiers and functions has been described in Section 1.4. If new candidates satisfy at least one of conditions and violate none of restrictions, they are inserted to the lexicon of corresponding POS tag. The conditions (items with [c]) and restrictions (items with [r]) of modifiers and functions are listed as follows.

Modifier For each POS tag with some category X, we insert new modifier candidate X/X (and corresponding X\X) to the right- (and corresponding left-) adjacent POS tag, if:

[c] X is an atomic category;

[c] X is a modifier itself.

Functor For adjacent POS tags with categories X and Y, we consider that X may take Y as argument to form the functor category X/Y, and Y may also take X as argument result in the functor Y\X. The new category is valid if the head H and argument A pass the following tests:

[c] H is modifier or in the form of (S|...), and A is atomic category N or S;

[c] H is S and A is N, i.e. categories S/N and S\N are allowed;

[c] A is not modifier, i.e. any non-modifier (atoms and functors) may be argument;

[r] H is different from A, otherwise the result category is modifier rather than functor;

[r] H is not N, since we assume that atomic N can not take any arguments.

After creating lexicon, we parse the sentences with CCG and remove categories that can not lead to a parse. The rest categories for POS tags are used to update the lexicon for each step. We perform this induction step twice to obtain the final lexicon $\mathcal{L}^{(2)}$.

The above induction procedure is almost the same as the algorithm described in (Bisk and Hockenmaier, 2012b). They also introduced an additional induction step to combine adjacent constituents that can be derived from the existing lexicon. However, their experiments did not show significant improvement of this “derived” lexicon generation step, so we omit this step in our experiments.

5.2 Improved CCG Induction Models

5.2.1 Basic Probabilistic Model

The *basic* model is the baseline model described in (Hockenmaier and Steedman, 2002), which is also used in (Bisk and Hockenmaier, 2012b). There are four types of rules in CCG: lexical (W) rules generate terminal words; unary (U) rules which could be root rules or type-raising rules; left-headed (L) rules with the first symbol as functor, e.g. the forward composition rules; and right-headed (R) rules with the second symbol as functor, e.g. the backward composition rules. Binary trees are generated top-down recursively from the special start symbol TOP. For each unexpanded nonterminal P, the basic model first generates the expansion type $\texttt{exp} \in \{\texttt{W}, \texttt{U}, \texttt{L}, \texttt{R}\}$ according to $P_e(\texttt{exp}|\texttt{P})$. Then for each expansion type, the model generates either terminal word w or head child H and possible non-head child N:

$$\begin{aligned}
\text{Lexical:} \quad & P_e(\texttt{exp} = \texttt{W}|\texttt{P})\, P_w(\texttt{w}|\texttt{P}, \texttt{exp} = \texttt{W}) \\
\text{Unary:} \quad & P_e(\texttt{exp} = \texttt{U}|\texttt{P})\, P_U(\texttt{H}|\texttt{P}, \texttt{exp} = \texttt{U}) \\
\text{Left:} \quad & P_e(\texttt{exp} = \texttt{L}|\texttt{P})\, P_L(\texttt{H}|\texttt{P}, \texttt{exp} = \texttt{L})\, P_l(\texttt{N}|\texttt{P}, \texttt{H}, \texttt{exp} = \texttt{L}) \\
\text{Right:} \quad & P_e(\texttt{exp} = \texttt{R}|\texttt{P})\, P_R(\texttt{H}|\texttt{P}, \texttt{exp} = \texttt{R})\, P_r(\texttt{N}|\texttt{P}, \texttt{H}, \texttt{exp} = \texttt{R})
\end{aligned}$$

where the subscripts $\{e, w, U, L, l, R, r\}$ represent different probability distributions.

After the lexicon generation step (presented in Section 5.1), each POS tag acquires a lexicon of CCG categories. These lexicons are used to parse the training corpus and CCG rules are created. For parameter estimation, we implement the Expectation Maximization (EM) algorithm is used to learn probabilities in the basic model. In the full EM, the Inside-Outside algorithm (Lari and Young, 1990) is used to collect the expected counts in the E-step of EM algorithm. We also implement the k-best EM described in (Bisk and Hockenmaier, 2012b), in which the expected counts are collected from k best parse trees.

5.2.2 Boundary Models

Boundary part-of-speech (POS) tags have been proven useful for detecting phrase boundaries in supervised setting (Xiong et al., 2010; He et al., 2010) and in unsupervised grammar induction (Golland et al., 2012; Huang et al., 2012). We introduce this idea to unsupervised combinatory categorial grammar induction. Since the POS tags are used as input of the induction system, we use the terms "boundary word" and "boundary POS tag" interchangeably in this chapter.

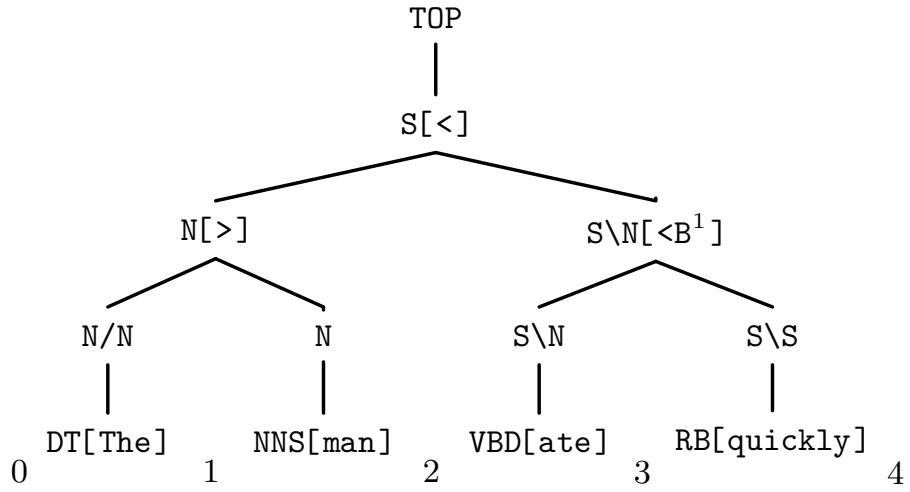


Figure 5.1: Illustration of the boundary probability calculation. The CCG rule types are given in the square brackets next to each nonterminal. Although only POS tags are considered in induction model, we also show the words for clarity.

Particularly, the boundary words of a given span are defined as the ordered pair of the leftmost and the rightmost POS tag of the constituent covered by the span. Given parse tree T, we define the new probabilistic model as

$$P(T) = \left(\prod_{\text{rule:}r \in T} P_{CCG}(r) \right) \left(\prod_{\text{span:}\langle i,j \rangle \in T} P_{BDR}(\sigma_{\langle i,j \rangle} | B) \right) \tag{5.1}$$

where distribution P_{CCG} is the basic CCG model defined in Section 5.2.1, P_{BDR} is the proposed boundary model, $\sigma_{\langle i,j \rangle}$ means the boundary POS tags of the constituent covered

by span $\langle i, j \rangle$, and B is a special nonterminal representing the constituent spans. Note the basic model P_{CCG} is defined over tree rules, and the boundary model P_{BDR} is defined over tree spans. This model is named as the *basic+bdr* model in experiments. The boundary model P_{BDR} could be learned by full EM and k-best EM, similar to the basic CCG model (subsection 5.2.1). We also propose Bayesian inference for both the baseline and boundary models (see next subsection).

Figure 5.1 shows an example of induced CCG tree. The probability of this parse tree under the boundary model is:

$$\begin{aligned} P_{BDR}(T) = & P(\texttt{DT\_DT}|\texttt{B}) \times P(\texttt{NNS\_NNS}|\texttt{B}) \times P(\texttt{VBD\_VBD}|\texttt{B}) \\ & \times P(\texttt{RB\_RB}|\texttt{B}) \times P(\texttt{DT\_NNS}|\texttt{B}) \times P(\texttt{VBD\_RB}|\texttt{B}) \\ & \times P(\texttt{DT\_RB}|\texttt{B}) \times P(\texttt{DT\_RB}|\texttt{B}) \end{aligned}$$

Note that the boundary probabilities are defined over the spans for each tree node, so for unary rules (e.g. the root rules and type-raising rules), the boundary probabilities may be calculated multiple times for the same span, e.g. the term $P(\texttt{DT\_RB}|\texttt{B})$ appears twice in the above example. This model is slightly different from the probability model of the constituent context model described in Section 4.1, in which the probabilities are defined over unique span set.

Currently, we use a single nonterminal `B` to represent all boundary tag pairs. We have also tried to let the boundary pairs depend on the category of corresponding tree nodes. For instance, the new boundary probability for the tree in Figure 5.1 becomes

$$\begin{aligned} P_{BDR}(T) = & P(\texttt{DT\_DT}|\texttt{N/N}) \times P(\texttt{NNS\_NNS}|\texttt{N}) \times P(\texttt{VBD\_VBD}|\texttt{S}\backslash\texttt{N}) \\ & \times P(\texttt{RB\_RB}|\texttt{S}\backslash\texttt{S}) \times P(\texttt{DT\_NNS}|\texttt{N}) \times P(\texttt{VBD\_RB}|\texttt{S}\backslash\texttt{N}) \\ & \times P(\texttt{DT\_RB}|\texttt{S}) \times P(\texttt{DT\_RB}|\texttt{TOP}) \end{aligned}$$

However, this category-dependent boundary model performs poor in experiments (not reported). The reason might be the data sparsity problem, since there are quite a lot of categories in the induced combinatory categorial grammar.

5.2.3 Bayesian Models

The EM algorithm may overfit the training data, so we propose the Bayesian model to infer grammars and tree structures. In Bayesian models, the generative process is often formulated as the Chinese Restaurant process (CRP) or the Pitman-Yor process (PYP) to encourage rule reuse and learn compact models (Teh et al., 2006; Pitman and Yor, 1997). Since PYP is a generation of CRP and has more elegant and controllable behaviour over the "long tail" of probability distributions, we focus on PYP in our approach.

The detailed PYP has been given in Section 3.1.2, and we apply the PYP into CCG induction. For each nonterminal A in CCG, we maintain a cache to store the total number n of rules expanded with A as parent, the total different rule types m, and the counts n_k of each rule that has been generated, for $k = 1, \ldots, m$. Initially, all caches are empty, i.e. with $n = m = 0$ and parse trees are generated in sequence. For each sentence, the PYP generates trees in top-down fashion. For each nonterminal label to be expanded, we consult the cache associated with that nonterminal and decide whether to choose the k^{th} rule in the cache, or generate a new rule. The probability of these two cases is

$$P_t(z|z_{i<n}) = \begin{cases} \frac{ma+b}{n+b}, & \text{if } z_{n+1} = m+1 \\ \frac{n_k - a}{n+b}, & \text{if } z_{n+1} = k, k \in \{1, \cdots, m\} \end{cases} \tag{5.2}$$

where z_i is the cache index of the i^{th} generated rule, $a \in [0, 1]$ and $b \geq 0$ are two label-associated parameters naming the discount and concentration parameters respectively. Note that different labels may have different values of a and b. If we decide to generate a new rule, then the new rule is sampled from the base multinomial distribution P_0. We also put a Dirichlet prior on the base distribution and sample the base rule probabilities $\boldsymbol{\theta} \sim \mathrm{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha})$. The above sampling procedures are performed recursively down until all frontier labels are terminals. For CCG induction models described in previous sections, PYP priors are put on all factored models, although they may have different hyperparameters.

To infer trees and parameters of PYP model, we apply the collapsed Metropolis-Hastings algorithm (Hastings, 1970; Johnson et al., 2007b) to sample trees from parse forest. In detail, we iteratively draw samples for each yield in training corpus in sequence or in random order. Assuming the current tree of the i^{th} POS tag sequence is T_i, we first remove this tree from the whole tree set to obtain $\mathcal{T}_{-i}$, the set of sampled trees except the i^{th} one. Then we draw new tree T_i' from some proposal distribution $Q(T_i'|\mathcal{T}_{-i})$, and accept the new sampled tree with probability

$$A(T_i, T_i') = \min\left\{1, \frac{P(T'|\alpha, \boldsymbol{a}, \boldsymbol{b})\ Q(T_i|\mathcal{T}_{-i})}{P(T|\alpha, \boldsymbol{a}, \boldsymbol{b})\ Q(T_i'|\mathcal{T}_{-i})}\right\}. \tag{5.3}$$

In theory, Q could be any distribution if it never assigns zero probability. In practice, the proposal distribution should be close enough to the true distribution to avoid high rejection rate. We use following proposal distribution in experiments:

$$Q(T_i|\mathcal{T}_{-i}) = \frac{1}{Z(\mathcal{T}_{-i})} \prod_{\text{rule:}r \in T_i} P_t(z_r|z_{\mathcal{T}_{-i}})\ P_0(r|\alpha)^{\delta(r \notin \mathcal{T}_{-i})} \tag{5.4}$$

in which P_t is the conditional index probability in Equation (5.2), and the model needs to consult the base distribution P_0 if it encounters a new rule ($\delta(r \notin \mathcal{T}_{-i}) = 1$). We do not need to calculate the normalization constant $Z(\mathcal{T}_{-i})$ since it would be cancelled in Equation (5.3). The proposal distribution differs from true distribution in the sense that caches are updated immediately after calculating probabilities of each rule in T_i under the true distribution, while the caches stay fixed in proposal distribution evaluation. In experiments, we observe that only a tiny fraction (less than 1%) of proposals are rejected. This provides evidence that the proposal distribution works well enough. We use the sampling algorithm described in (Blunsom and Osborne, 2008) to draw a parse tree from the parse forest according to the proposal distribution Q.

5.3 Experiments

5.3.1 Datasets and Settings

We carry out experiments on the Wall Street Journal portion of the Penn English Treebank (Marcus et al., 1993). As the standard data split, we use sections 02-21 as the training set, section 00 as the development set, and section 23 as the final test set. We remove punctuations and null elements in treebank, as the standard preprocessing step in previous unsupervised grammar induction approaches (Klein and Manning, 2002; Cohn et al., 2010; Bisk and Hockenmaier, 2012b). For comparison, we build datasets with sentence lengths no more than 10, 20, 30 and 40 words after removing punctuations.

Dataset	Train		Dev		Test	
	# sent	# word	# sent	# word	# sent	# word
PTB10	5899	41701	265	1875	398	2649
PTB20	-	-	-	-	1286	16591
PTB30	-	-	-	-	2028	35148
PTB40	-	-	-	-	2338	45813

Table 5.1: Penn treebank data statistics.

As the standard machine learning pipeline, we perform learning and inference on the training set, select model with best performance on the development set, and report the result of selected model on the test set. Theoretically, we should tune and test parameters on corpora with the same length. However, the number of CCG categories obtained is huge, so it takes quite a long time on tuning parameters on long sentences. As a result, following previous approach (Bisk and Hockenmaier, 2012b), we only train and tune parameters on sentences with length no more than 10, but report performance on longer sentences as well. Table 5.1 gives the statistics for each dataset.

The original Penn treebank only has constituency trees, but we evaluate the performance of dependency trees. Converting constituency trees to dependency trees is not

a trivial process, in which the head word of each constituent tree node must be identified. This is usually done using manually written converting rules (Collins, 1999). To be consistent with previous work, we use (Johansson and Nugues, 2007)'s code[3] to convert treebank to dependency structures. Figure 5.2 and 5.3 show a constituency tree and the corresponding converted dependency tree.

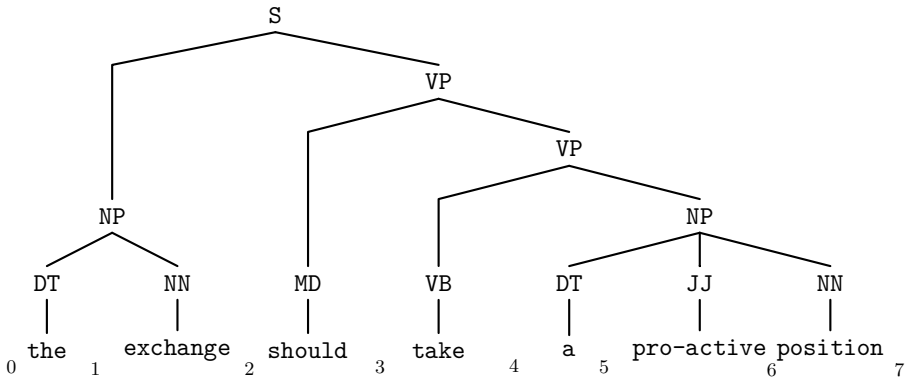


Figure 5.2: An example of constituency reference tree.

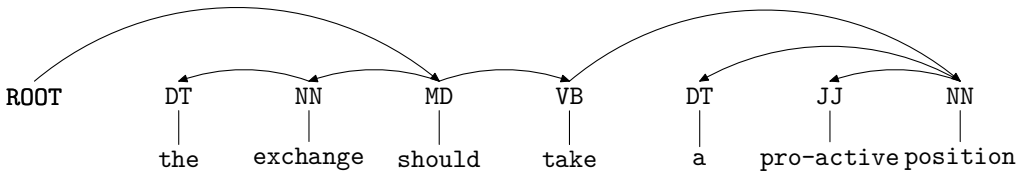


Figure 5.3: An example of converted reference dependency structure.

For natural languages, most dependencies are between adjacent words, such as the adjacent adjectives and nouns. Similar to the trivial left- and right-branching baseline in constituency grammar induction (Section 4.3), we also investigate two trivial baseline, named backward linked tree and forward linked tree, for dependency induction system. Figure 5.4 shows the backward linked dependency structure (corresponding to the left-branching constituency tree), in which each word takes the preceding word as an argument and the last word acts as the head of the whole sentence. Figure 5.5 shows the

[3] http://nlp.cs.lth.se/software/treebank_converter

forward linked dependency structure (corresponds to the right-branching constituency tree), where each word is the head of the succeeding word and the first word links to the root. We report evaluation results for these trivial baselines in experiments as well.

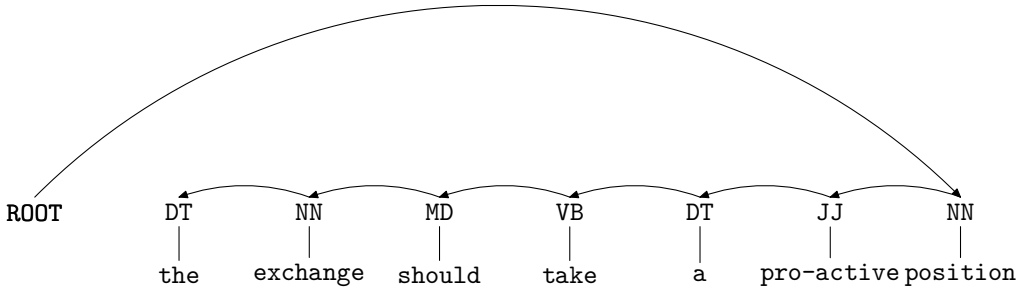


Figure 5.4: An example of backward-linked dependency structure.

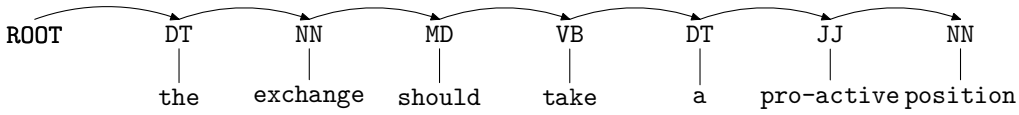


Figure 5.5: An example of forward-linked dependency structure.

To reduce model complexity, we restrict that the maximal order of composition rule is 2. The rule probabilities are initialized uniformly. For full EM models, we add fixed value to expected counts in each E-step as smoothing. We perform maximal 40 EM iterations while stop earlier if the development score starts to drop. For k-best EM models, we interpolate the k-best probabilities and the full probabilities as described in the footnote in (Bisk and Hockenmaier, 2012b). We test different k (number of best trees) and select the best one that achieving the best UAS in the development set. In the Bayesian inference, we run sampler through the whole training sentences for 400 iterations and use the last sampled grammars to parse fresh sentences. Following (Johnson and Goldwater, 2009; Huang et al., 2011), we put an uninformative $\mathrm{Beta}(1, 1)$ prior on $\boldsymbol{a}$ and a "vague" $\mathrm{Gamma}(10, 0.1)$ prior on $\boldsymbol{b}$ to model the uncertainty of these hyperparameters. After each iteration, we resample each of hyperparameters from the posterior distribution of hyperparameters using a slice sampler (Neal, 2003).

5.3.2 Evaluation Metrics

For the induced CCG trees, we follow (Bisk and Hockenmaier, 2012b; Bisk and Hockenmaier, 2012a) to convert CCG trees to dependency trees: (1) modifiers are treated as dependents of their heads; (2) the head of the sentence is treated as a dependent of a special root node at position 0; (3) the left part of conjunction is treated as the head of `conj`, and `conj` is treated as the head of right part. Figure 5.6 and 5.7 show an example of the induced tree of combinatory categorial grammar and the corresponding converted dependency structure. The dependency links (represented as arrows) are pointed from the head word to its arguments. Note that the dependencies are unlabeled, since we do not have label information in unsupervised induction.

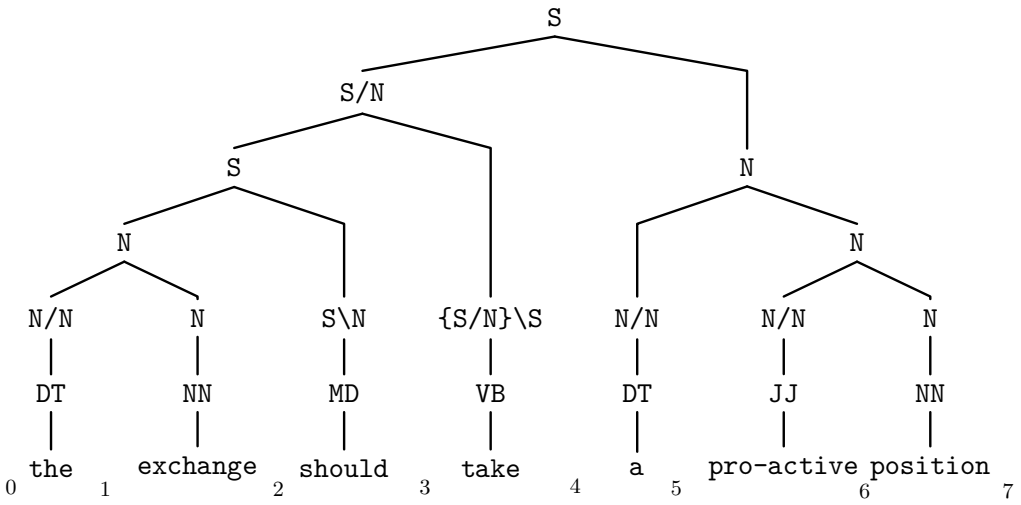


Figure 5.6: An example of constituency candidate tree.

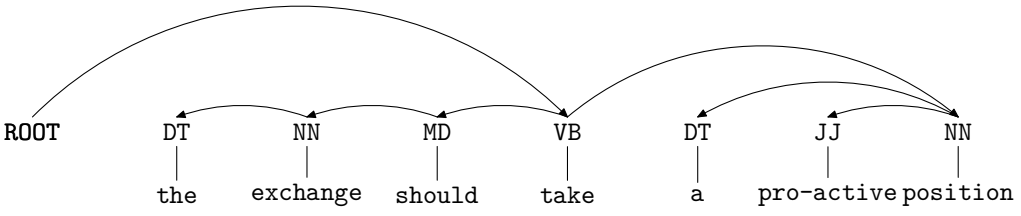


Figure 5.7: An example of converted candidate dependency structures.

For evaluation, we represent a dependency tree as a set of dependency links. There are always n dependency links for sentences with length n . Therefore, there is no difference between precision and recall, and we measure dependency accuracy straightforwardly by comparing the two dependency link sets of reference and candidate dependency trees. The accuracy can be evaluated for the directed or undirected links, in which the former one consider the link directions but the latter one ignore the link directions. We adopt the directed accuracy and use the script of CoNLL 2008 shared task[4] to calculate the Unlabeled Attachment Score (UAS) . Note that the UAS is calculated over the whole dataset rather than individual sentences. We perform the McNemar's significant test (McNemar, 1947) to compare the proposed models with the baseline models.

We show an evaluation example here. The reference tree in Figure 5.3 and the candidate tree in Figure 5.7 can be represented as following directed link sets

Ref	Cand	Matched
$\langle$[0]ROOT, [3]MD, $\rightarrow\rangle$	-	-
$\langle$[1]DT, [2]NN, $\leftarrow\rangle$	$\langle$[1]DT, [2]NN, $\leftarrow\rangle$	$\langle$[1]DT, [2]NN, $\leftarrow\rangle$
$\langle$[2]NN, [3]MD, $\leftarrow\rangle$	$\langle$[2]NN, [3]MD, $\leftarrow\rangle$	$\langle$[2]NN, [3]MD, $\leftarrow\rangle$
$\langle$[3]MD, [4]VB, $\rightarrow\rangle$	-	-
$\langle$[4]VB, [7]NN, $\rightarrow\rangle$	$\langle$[4]VB, [7]NN, $\rightarrow\rangle$	$\langle$[4]VB, [7]NN, $\rightarrow\rangle$
$\langle$[5]DT, [7]NN, $\leftarrow\rangle$	$\langle$[5]DT, [7]NN, $\leftarrow\rangle$	$\langle$[5]DT, [7]NN, $\leftarrow\rangle$
$\langle$[6]JJ, [7]NN, $\leftarrow\rangle$	$\langle$[6]JJ, [7]NN, $\leftarrow\rangle$	$\langle$[6]JJ, [7]NN, $\leftarrow\rangle$
-	$\langle$[0]ROOT, [4]VB, $\rightarrow\rangle$	-
-	$\langle$[3]MD, [4]VB, $\leftarrow\rangle$	-
$\lvert\mathcal{G}\rvert = \lvert\mathcal{C}\rvert = 7$		$\lvert\mathcal{M}\rvert = 5$

As a result, the unlabeled attach score for this example are $\frac{5}{7}$. In the similar way, the UAS of the corresponding backward linked tree (Figure 5.4) and forward linked tree (Figure 5.5) are $\frac{3}{7}$ and $\frac{1}{7}$ respectively.

[4] http://barcelona.research.yahoo.net/dokuwiki/doku.php?id=conll2008:software

5.3.3 Smoothing Effects in Full EM Models

We first carry out experiments to examine the effect of smoothing values for full EM models. We test smoothing values from {1,10,20,30,40,50,60,70,80,90,100} and evaluate the unlabeled attachment scores (UAS) of basic model and basic+bdr model on the development set and the PTB10 test set. Note that the final smoothing value is selected as the one with best performance on the development set (not related to the test set), and the results on the PTB10 test set are only given as a reference.

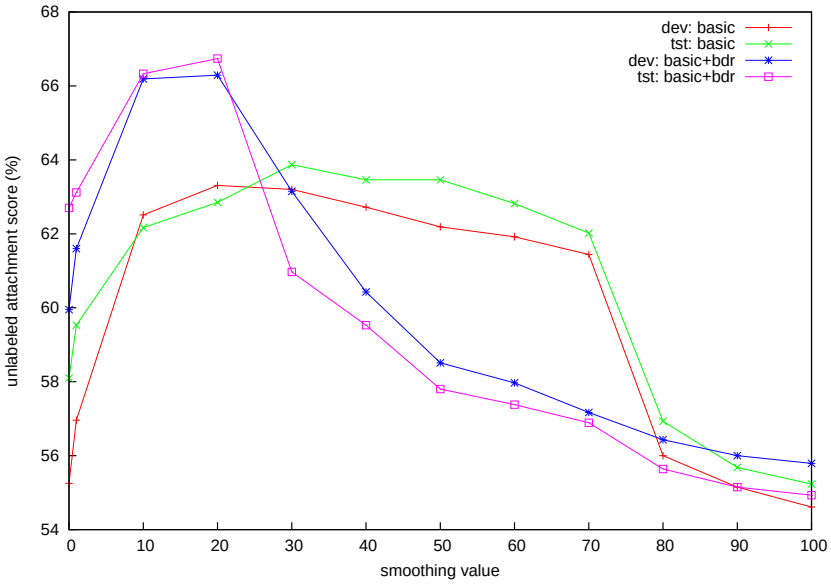


Figure 5.8: Impact of smoothing values on full EM learning for CCG induction. The dependency accuracy values on the development and test set of PTB10 are plotted.

The experimental results are plotted in Figure 5.8. The accuracy scores on the development set first increase then decrease with the increment of smoothing value. We can easily find that the best smoothing value (with highest dev-score) is 20 for both the basic model and basic+bdr model. The basic+bdr model achieves significant better results (dev: 66.3, tst: 66.7) than the basic model (dev: 63.3, tst: 62.9) at $p < 10^{-3}$ level on both development and test set when optimal smoothing values are selected.

5.3.4 K-best EM vs. Full EM

In k-best EM, we select k from {1, 10, (step 10), 200, (step 20), 300, (step 100), 1000}. Note that when $k = 1$, the 1-best learning is known as the Viterbi learning algorithm. The unlabeled attachment scores of k-best EM on the development set of basic and basic+bdr models for different values of k are plotted in Figure 5.9. The best results of full EM are also shown for comparison.

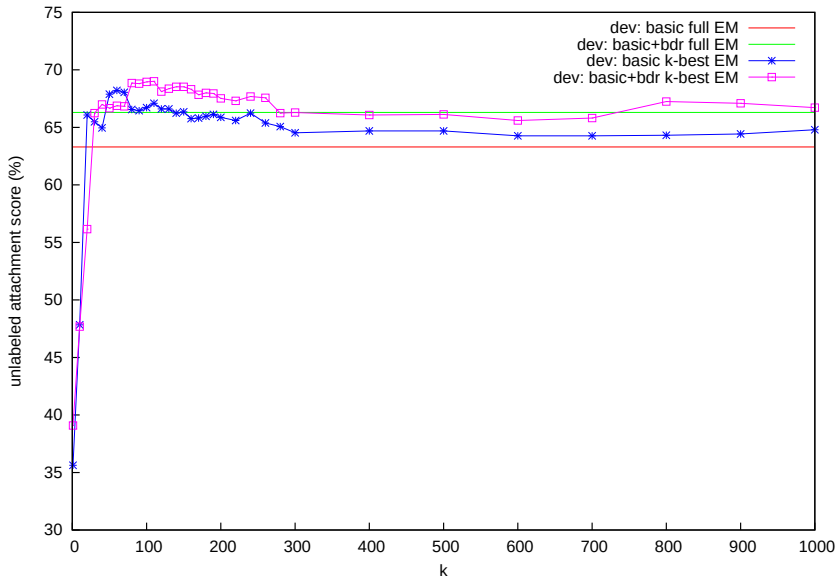


Figure 5.9: Impact of k on k-best EM learning for CCG induction. The dependency accuracy values on the development set of PTB10 are plotted.

From this figure, we can see that the accuracy scores of k-best models increase quickly with the increment of k, then decrease slowly and finally converge to some steady points. Secondly, the best results of k-best EM exceed the full EM, proved the conclusion in (Bisk and Hockenmaier, 2012b). Thirdly, the basic+bdr models outperform the basic models, which demonstrates the effectiveness of boundary words. Finally, the Viterbi results are lower than both the results of full EM and k-best EM, which is consistent with (Bisk and Hockenmaier, 2012b), but opposite to (Spitkovsky et al., 2010b).

5.3.5 Induction Results

The final results over all datasets are shown in Table 5.2 for comparison. We report the unlabeled attachment scores for the basic and basic+bdr models using full EM, k-best EM and Pitman-Yor process (PYP) as learning methods. Some results of existing approaches are included in this table as well.

Model		PTB10	PTB20	PTB30	PTB40
(Klein and Manning, 2004)		47.5	-	-	-
(Headden III et al., 2009)		68.8	-	-	-
(Spitkovsky et al., 2010b)		65.3*	53.8*	-	-
(Cohn et al., 2010)		65.9	58.3	-	-
(Bisk and Hockenmaier, 2012b)		71.5	60.3	-	-
(Naseem et al., 2010)		71.9	50.4*	-	-
Trivial	backward linked	32.7	28.8	27.7	27.2
	forward linked	25.4	25.7	26.3	26.4
Viterbi EM	basic	39.2	23.2	18.5	16.8
	basic+bdr	39.0	27.0	23.2	22.0
k-best EM	basic	67.3	56.0	52.0	50.4
	basic+bdr	**68.1**	**56.6**	**52.8**	**51.4**
full EM	basic	62.9	49.9	46.0	44.6
	basic+bdr	66.7	54.0	49.4	48.2
PYP	basic	66.0	53.9	50.5	48.8
	basic+bdr	66.7	55.1	51.0	49.0

Table 5.2: Induction results of improved CCG models. Results of existing approaches are copied from (Bisk and Hockenmaier, 2012b). Starred results were obtained with additional training data.

From this table, we can see that both the trivial backward linked and forward linked baselines perform poor at the evaluation of dependency accuracy. These results are quite different from the constituency grammar induction results (shown in Section 4.3.3), in which the trivial right-branching constituency trees achieve good performance. This could be explained that although the constituency trees are right-branching preferable, the head words of constituents have not left or right position preference.

Viterbi EM can be seen as a special case of k-best EM with $k = 1$. Although Spitkovsky et al. (2010b) demonstrate the Viterbi training method improves the Dependency Model with Valence (DMV), it does not perform well for our CCG induction model. Experimental results show that with a suitable selection of k, the k-best EM outperforms the full EM, which is consistent with (Bisk and Hockenmaier, 2012b).

With the introduction of Pitman-Yor prior distributions, the proposed Bayesian inference improves the full EM induction results. This provides evidence that compact models are preferred in unsupervised CCG induction. Lower than expected, however, the PYP results are still lower than the k-best EM results. The reason might be the k-best EM is more likely to escape from local optima, while the sampling procedure needs too many iterations to converge and usually gets stuck in local optima in practice.

Boundary models (basic+bdr) consistently show better performance than the corresponding baseline models (basic), for all the full EM, k-best EM and Bayesian learning models. The improvements of boundary models under Bayesian inference is relatively smaller than the full EM and k-best EM. The reason might be that both the boundary models and Bayesian models give high probabilities to those parse trees with more reused rules, so the combination of them only performs slightly better than individual component. For longer sentences, the boundary methods still outperform baseline model, demonstrating the robustness of our method.

Compared with existing approaches in Table 5.2, our models stay in the intermediate level. The dependency accuracy scores of the Dependency Model with Valence (DMV) (Klein and Manning, 2004) are much lower than ours. Headden III et al. (2009) improve the basic DMV using rich contexts, words as well as POS tags, and sophisticated smoothing techniques, which might explain their higher performance than ours on short sentences. Spitkovsky et al. (2010b) propose to use the Viterbi learning for DMV, but their results are lower than our reported ones. Cohn et al. (2010) propose complicated Bayesian models for the tree-subsection grammars, which is difficult to implement

and tune. Naseem et al. (2010) manually specify some dependency rules in experiments, while we just use some coarse restrictions on lexicon and grammar generation. Bisk and Hockenmaier (2012b) reports better results than our models on both short and long sentences. Our basic models with full EM and k-best EM are the reimplementation of their models. As their induction codes are not public available, however, we may miss some details in implementation, and can not reproduce their results.

5.3.6 Discussion

Our method takes the golden part-of-speech tags as input. This practice may reduce data sparsity problem caused by directly modeling words. However, this may also lose useful lexical information. As reported in (Headden III et al., 2009), incorporating words with high frequencies (greater than 100 times in their experiments) as well as the POS tags could improve the induction accuracy for dependency models. In CCG, words may also help to distinguish lexical categories. For example, the transitive verbs are often tagged as (S\N)/N and the intransitive verbs often have category S\N. However, these syntactic differences are not encoded in the Penn treebank POS tags, in which they may both have the POS tag VBx depending on the tenses. How to use rich lexical information to help the CCG induction is one possible research direction of our work.

Although the simple additive smoothing methods could improve EM results (see Figure 5.8), sophisticated smoothing schemes are also applicable (Headden III et al., 2009). Currently, the final probability is the product of basic CCG model and boundary model, which is motivated by the agreement measurement in word alignment (Liang et al., 2006). Although this simple strategy has already shown effectiveness in our experiments, other interpolation techniques could be also tested. In addition, the context POS tags have been proved useful for constituency tree induction (Klein and Manning, 2002; Golland et al., 2012; Huang et al., 2012). Using context information is another extension of our current work.

5.4 Summary

In this chapter, we have proposed to incorporate lexical information in unsupervised CCG induction. Specifically, an additional boundary model is defined to capture complex language aspects, in which boundary words are generated from a special symbol independently for each span covered by tree nodes. Furthermore, we describe nonparametric Pitman-Yor process to encourage rule reuse. Experimental results demonstrate that the boundary models consistently improve the baseline models for all learning algorithms and over all datasets. The Bayesian inference outperforms the full EM, but still underperforms the k-best EM.

Chapter 6

Conclusion

6.1 Summary of Achievements

In this dissertation, we focus on three unsupervised structure induction problems: the transliteration equivalence learning, the constituency grammar induction and the dependency grammar induction. We make following contributions:

- We review the overfitting problem of existing EM-based transliteration models and propose a general nonparametric Bayesian learning framework for transliteration. We demonstrate how to represent the syllable learning problem as the grammar inference problem. The proposed synchronous adaptor grammars (SAGs) could automatically discover syllable equivalents without any heuristics or restrictions. The joint source-channel model is then used for training and decoding. Experimental results on transliteration task of four language pairs show that the proposed method outperforms the EM-based baseline system. In this point of view, the new model is language independent.

- We discuss the problems of constituent-context model (CCM) for constituency grammar induction and present the feature-based CCM in which linguistic knowledge could be easily incorporated. The EM algorithm is still applicable for this

local normalization method. The use of ℓ_1-norm regularization leads to compact grammars. We also propose a reasonable model selection and evaluation strategy. Experiments demonstrate that the presented model achieves comparable performance on the short sentences but significant improvements on the longer sentences.

- We investigate the state-of-the-art combinatory categorial grammar (CCG) induction approach and propose to use boundary part-of-speech tags and Bayesian learning to improve the EM baseline. Specifically, an additional boundary model is defined to capture constituents, in which boundary words are generated from a special symbol independently for each span covered by tree nodes. We also propose a Bayesian model based the Pitman-Yor process to encourage rule reuse. The full EM and k-best EM learning algorithms are also implemented for comparison. Experimental results demonstrate that the boundary models consistently improve the baseline models for all learning algorithms and over all datasets. The Bayesian inference outperforms the full EM, but the k-best EM performs the best.

6.2 Future Directions

In this dissertation, sampling techniques are used to infer grammars for Bayesian models (see Chapter 3 and 5), since they are easy to implement. Although correct sampling implementations guarantee to converge to the real probability distributions, the converging speed is often slow in practice. An alternative approximating inference technique is the variational Bayesian inference, which casts the posterior inference as a deterministic optimization problem (Jordan et al., 1999; Cohen et al., 2010).

Currently, we use the joint source-channel model as the decoding model for transliteration. Similar the probabilistic inference for machine translation (Blunsom and Osborne, 2008), we can also directly use the synchronous adaptor grammars as decoding models, instead of converting the inferred grammars to lattice and then using the joint source-

channel model to decode.

For feature-based CCM, we only experiment a few feature templates. Other features such as words, stems may improve the performance. Moreover, punctuations are useful information in grammar induction (Spitkovsky et al., 2011b; Ponvert et al., 2011), while currently punctuations are ignored in our model.

The lexicon generation step is very important for the CCG induction. In this thesis, we just follow previous work (Bisk and Hockenmaier, 2012b) to automatically generate lexicons for each part-of-speech tag from the basic categories `S` and `N`. We may assign more linguistic-motivated initial categories (Watkinson and Manandhar, 1999) to the induction system.

Another direction is to use induced structures in subsequent NLP tasks, e.g. machine translation. One issue should be mentioned is that the evaluation metrics used in unsupervised learning tasks are different from the final evaluation metrics used for application tasks. For example, the treebank F_1 score is used to evaluate the constituency tree induction system, while the BLEU (Papineni et al., 2002) is commonly used to evaluate machine translation. We may use the final evaluation metric to guide the induction task.

Bibliography

[Andrew and Gao2007] Galen Andrew and Jianfeng Gao. 2007. Scalable training of l1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40, Corvalis, Oregon, USA, June.

[Aramaki and Abekawa2009] Eiji Aramaki and Takeshi Abekawa. 2009. Fast decoding and easy implementation: Transliteration as sequential labeling. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 65–68, Suntec, Singapore, August.

[Berg-Kirkpatrick et al.2010] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California, June.

[Bisk and Hockenmaier2012a] Yonatan Bisk and Julia Hockenmaier. 2012a. Induction of linguistic structure with combinatory categorial grammars. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 90–95, Montréal, Canada, June.

[Bisk and Hockenmaier2012b] Yonatan Bisk and Julia Hockenmaier. 2012b. Simple robust grammar induction with combinatory categorial grammar. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 1643–1649, Toronto, Canada, July.

[Black et al.1991] E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the Workshop on Speech and Natural Language*, pages 306–311, Pacific Grove, California, February.

[Blunsom and Cohn2010] Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213, Cambridge, MA, October.

[Blunsom and Osborne2008] Phil Blunsom and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 215–223, Honolulu, Hawaii, October.

[Bod1998] Rens Bod. 1998. *Beyond Grammar: an experience-based theory of language*.

[Bod2003] Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 19–26, Budapest, Hungary, April.

[Bod2006a] Rens Bod. 2006a. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872, Sydney, Australia, July.

[Bod2006b] Rens Bod. 2006b. Unsupervised parsing with U-DOP. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 85–92, New York City, June.

[Bod2007] Rens Bod. 2007. Is the end of supervised parsing in sight? In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 400–407, Prague, Czech Republic, June.

[Boonkwan and Steedman2011] Prachya Boonkwan and Mark Steedman. 2011. Grammar induction from text using small syntactic prototypes. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 438–446, Chiang Mai, Thailand, November.

[Brown et al.1993] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June.

[Charniak2000] Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, pages 132–139, Seattle, Washington.

[Clark2001] Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning*, Toulouse, France, July.

[Clark2003] Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 59–66, Budapest, Hungary, April.

[Cocke and Schwartz1970] John Cocke and Jacob T. Schwartz. 1970. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University.

[Cohen and Smith2009] Shay Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82, Boulder, Colorado, June.

[Cohen et al.2010] Shay B. Cohen, David M. Blei, and Noah A. Smith. 2010. Variational inference for adaptor grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 564–572, Los Angeles, California, June.

[Cohn and Blunsom2010] Trevor Cohn and Phil Blunsom. 2010. Blocked inference in bayesian tree substitution grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 225–230, Uppsala, Sweden, July.

[Cohn et al.2009] Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556, Boulder, Colorado, June.

[Cohn et al.2010] Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing Tree-Substitution grammars. *Journal of Machine Learning Research*, 11:3053–3096.

[Collins1997] Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, July.

[Collins1999] Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

[DeNero and Uszkoreit2011] John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Edinburgh, Scotland, UK., July.

[DeNero et al.2008] John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 314–323, Honolulu, Hawaii, October.

[Dyer et al.2011] Chris Dyer, Jonathan H. Clark, Alon Lavie, and Noah A. Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 409–419, Portland, Oregon, USA, June.

[Earley1983] Jay Earley. 1983. An efficient context-free parsing algorithm. *Communications of the ACM*, 26(1):57–61, January.

[Eisner1996] Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86, Santa Cruz, California, USA, June.

[Finch and Sumita2008] Andrew Finch and Eiichiro Sumita. 2008. Phrase-based machine transliteration. In *Proceedings of the Workshop on Technologies and Corpora for Asia-Pacific Speech Translation (TCAST)*, pages 13–18, Hyderabad, India, January.

[Finch and Sumita2009] Andrew Finch and Eiichiro Sumita. 2009. Transliteration by bidirectional statistical machine translation. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 52–56, Suntec, Singapore, August.

[Finch and Sumita2010a] Andrew Finch and Eiichiro Sumita. 2010a. A Bayesian model of bilingual segmentation for transliteration. In *Proceedings of the 7th International Workshop on Spoken Language Translation*, pages 259–266, Paris, France, December.

[Finch and Sumita2010b] Andrew Finch and Eiichiro Sumita. 2010b. Transliteration using a phrase-based statistical machine translation system to re-score the output of a joint multigram model. In *Proceedings of the 2010 Named Entities Workshop*, pages 48–52, Uppsala, Sweden, July.

[Finkel et al.2007] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2007. The infinite tree. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 272–279, Prague, Czech Republic, June.

[Freitag and Wang2009] Dayne Freitag and Zhiqiang Wang. 2009. Name transliteration with bidirectional perceptron edit models. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 132–135, Suntec, Singapore, August.

[Golland et al.2012] Dave Golland, John DeNero, and Jakob Uszkoreit. 2012. A feature-rich constituent context model for grammar induction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 17–22, Jeju Island, Korea, July.

[Goodman1996] Joshua Goodman. 1996. Efficient algorithms for parsing the DOP model. In *Proceedings of the 1996 Conference on Empirical Methods in Natural Language Processing*, pages 143–152, Philadelphia, Pennsylvania, May.

[Hardisty et al.2010] Eric Hardisty, Jordan Boyd-Graber, and Philip Resnik. 2010. Modeling perspective using adaptor grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 284–292, Cambridge, MA, October.

[Hastings1970] W. K. Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

[He et al.2010] Zhongjun He, Yao Meng, and Hao Yu. 2010. Learning phrase boundaries for hierarchical phrase-based translation. In *Coling 2010: Posters*, pages 383–390, Beijing, China, August.

[Headden III et al.2009] William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June.

[Hockenmaier and Bisk2010] Julia Hockenmaier and Yonatan Bisk. 2010. Normal-form parsing for combinatory categorial grammars with generalized composition and type-raising. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 465–473, Beijing, China, August.

[Hockenmaier and Steedman2002] Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Philadelphia, Pennsylvania, USA, July.

[Hockenmaier and Steedman2007] Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396, September.

[Hong et al.2009] Gumwon Hong, Min-Jeong Kim, Do-Gil Lee, and Hae-Chang Rim. 2009. A hybrid approach to english-korean name transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 108–111, Suntec, Singapore, August.

[Hopcroft et al.2006] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation*. Boston, MA, USA, 3rd edition.

[Huang et al.2011] Yun Huang, Min Zhang, and Chew Lim Tan. 2011. Nonparametric bayesian machine transliteration with synchronous adaptor grammars. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539, Portland, Oregon, USA, June.

[Huang et al.2012] Yun Huang, Min Zhang, and Chew Lim Tan. 2012. Improved constituent context model with features. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 564–573, Bali, Indonesia, November.

[Huang2009] Fei Huang. 2009. Confidence measure for word alignment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 932–940, Suntec, Singapore, August.

[Jansche and Sproat2009] Martin Jansche and Richard Sproat. 2009. Named entity transcription with pair n-gram models. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 32–35, Suntec, Singapore, August.

[Jia et al.2009] Yuxiang Jia, Danqing Zhu, and Shiwen Yu. 2009. A noisy channel model for grapheme-based machine transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 88–91, Suntec, Singapore, August.

[Jiang et al.2009] Xue Jiang, Le Sun, and Dakun Zhang. 2009. A syllable-based name transliteration system. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 96–99, Suntec, Singapore, August.

[Johansson and Nugues2007] Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, pages 105–112, Tartu, Estonia, May.

[Johnson and Demuth2010] Mark Johnson and Katherine Demuth. 2010. Unsupervised phonemic chinese word segmentation using adaptor grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 528–536, Beijing, China, August.

[Johnson and Goldwater2009] Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June.

[Johnson et al.2007a] Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007a. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April.

[Johnson et al.2007b] Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in Neural Information Processing Systems 19*, pages 641–648, Cambridge, MA.

[Johnson2002] Mark Johnson. 2002. The DOP estimation method is biased and inconsistent. *Computational Linguistics*, 28(1):71–76, March.

[Johnson2008] Mark Johnson. 2008. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of ACL-08: HLT*, pages 398–406, Columbus, Ohio, June.

[Johnson2010] Mark Johnson. 2010. PCFGs, topic models, adaptor grammars and learning topical collocations and the structure of proper names. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1157, Uppsala, Sweden, July.

[Jones et al.2010] Bevan K. Jones, Mark Johnson, and Michael C. Frank. 2010. Learning words and their meanings from unsegmented child-directed speech. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 501–509, Los Angeles, California, June.

[Jordan et al.1999] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November.

[Joshi and Schabes1997] Aravind K. Joshi and Yves Schabes, 1997. *Handbook of Formal Languages, vol. 3: beyond words*, chapter 2, pages 69–124. New York, NY, USA.

[Khapra and Bhattacharyya2009] Mitesh Khapra and Pushpak Bhattacharyya. 2009. Improving transliteration accuracy using word-origin detection and lexicon lookup. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 84–87, Suntec, Singapore, August.

[Klein and Manning2001] Dan Klein and Christopher D. Manning. 2001. Distributional phrase structure induction. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning*, Toulouse, France, July.

[Klein and Manning2002] Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA, July.

[Klein and Manning2004] Dan Klein and Christopher Manning. 2004. 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.

[Klein2005] Dan Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University.

[Knight and Graehl1998] Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612, December.

[Koehn et al.2003] Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 48–54, Edmonton, Canada, May.

[Lari and Young1990] K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

[Lewis II and Stearns1968] P. M. Lewis II and R. E. Stearns. 1968. Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488, July.

[Li et al.2004] Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 159–166, Barcelona, Spain, July.

[Li et al.2007] Haizhou Li, Khe Chai Sim, Jin-Shea Kuo, and Minghui Dong. 2007. Semantic transliteration of personal names. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 120–127, Prague, Czech Republic, June.

[Li et al.2009a] Haizhou Li, A. Kumaran, Vladimir Pervouchine, and Min Zhang. 2009a. Report of news 2009 machine transliteration shared task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 1–18, Suntec, Singapore, August.

[Li et al.2009b] Haizhou Li, A. Kumaran, Min Zhang, and Vladimir Pervouchine. 2009b. Whitepaper of news 2009 machine transliteration shared task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 19–26, Suntec, Singapore, August.

[Li et al.2012] Zhonghua Li, Jun Lang, Yun Huang, and Jiajun Chen. 2012. Feature-based itg for unsupervised word alignment. In *Proceedings the International Conference on Network and Computational Intelligence (ICNCI)*, Hong Kong, August.

[Liang and Klein2008] Percy Liang and Dan Klein. 2008. Analyzing the errors of unsupervised learning. In *Proceedings of ACL-08: HLT*, pages 879–887, Columbus, Ohio, June.

[Liang and Klein2009] Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 611–619, Boulder, Colorado, June.

[Liang et al.2006] Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 104–111, New York City, USA, June.

[Liang et al.2007] Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697, Prague, Czech Republic, June.

[Liu et al.2005] Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 459–466, Ann Arbor, Michigan, June.

[Liu et al.2009] Yang Liu, Tian Xia, Xinyan Xiao, and Qun Liu. 2009. Weighted alignment matrices for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1017–1026, Singapore, August.

[Marcus et al.1993] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, June.

[McNemar1947] Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, June.

[Merialdo1994] Bernard Merialdo. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–171, June.

[Mermer and Saraclar2011] Coskun Mermer and Murat Saraclar. 2011. Bayesian word alignment for statistical machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 182–187, Portland, Oregon, USA, June.

[Mirroshandel and Ghassem-Sani2008] Seyed Abolghasem Mirroshandel and Gholamreza Ghassem-Sani. 2008. Unsupervised grammar induction using a parent based constituent context model. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 293–297, Patras, Greece, July.

[Moore et al.2006] Robert C. Moore, Wen-tau Yih, and Andreas Bode. 2006. Improved discriminative bilingual word alignment. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 513–520, Sydney, Australia, July.

[Moore2004] Robert C. Moore. 2004. Improving IBM word alignment model 1. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 518–525, Barcelona, Spain, July.

[Nabende2009] Peter Nabende. 2009. Transliteration system using pair hmm with weighted fsts. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 100–103, Suntec, Singapore, August.

[Naseem et al.2010] Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, MA, October.

[Neal2003] Radford M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31(3):705–767.

[Nocedal1980] Jorge Nocedal. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782.

[Oh et al.2009] Jong-Hoon Oh, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Machine transliteration using target-language grapheme and phoneme: Multi-engine transliteration approach. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 36–39, Suntec, Singapore, August.

[Osborne and Briscoe1997] Miles Osborne and Ted Briscoe. 1997. Learning stochastic categorial grammars. In *Proceedings of CoNLL97: Computational Natural Language Learning*, pages 80–87.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

[Pervouchine et al.2009] Vladimir Pervouchine, Haizhou Li, and Bo Lin. 2009. Transliteration alignment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL*

and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 136–144, Suntec, Singapore, August.

[Pitman and Yor1997] J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900.

[Pitman1995] Jim Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability Theory Related Fields*, 102(2):145–158.

[Ponvert et al.2011] Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Portland, Oregon, USA, June.

[Ponvert2007] Elias Ponvert. 2007. Inducing combinatory categorial grammars with genetic algorithms. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 7–12, Prague, Czech Republic, June.

[Post and Gildea2009] Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48, Suntec, Singapore, August.

[Rama and Gali2009] Taraka Rama and Karthik Gali. 2009. Modeling machine transliteration as a phrase based statistical machine translation problem. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 124–127, Suntec, Singapore, August.

[Reddy and Waxmonsky2009] Sravana Reddy and Sonjia Waxmonsky. 2009. Substring-based transliteration with conditional random fields. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 92–95, Suntec, Singapore, August.

[Sangati and Zuidema2011] Federico Sangati and Willem Zuidema. 2011. Accurate parsing with compact tree-substitution grammars: Double-DOP. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 84–95, Edinburgh, Scotland, UK., July.

[Schütze1995] Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*, pages 141–148, Dublin, Ireland, March.

[Seginer2007] Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, Prague, Czech Republic, June.

[Shishtla et al.2009] Praneeth Shishtla, Surya Ganesh Veeravalli, Sethuramalingam Subramaniam, and Vasudeva Varma. 2009. A language-independent transliteration schema using character aligned models at news 2009. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 40–43, Suntec, Singapore, August.

[Skut et al.1998] Wojciech Skut, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1998. A linguistically interpreted corpus of german newspaper text. In *Proceedings of the European Summer School in Logic, Language and Information Workshop on Recent Advances in Corpus Annotation*, Saarbrücken, Germany.

[Smith and Eisner2004] Noah A. Smith and Jason Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 486–493, Barcelona, Spain, July.

[Smith and Eisner2005] Noah A. Smith and Jason Eisner. 2005. 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.

[Smith and Eisner2006] Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 569–576, Sydney, Australia, July.

[Smith and Johnson2007] Noah A. Smith and Mark Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–492, December.

[Spitkovsky et al.2010a] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California, June.

[Spitkovsky et al.2010b] Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17, Uppsala, Sweden, July.

[Spitkovsky et al.2011a] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011a. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1280, Edinburgh, Scotland, UK., July.

[Spitkovsky et al.2011b] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011b. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 19–28, Portland, Oregon, USA, June.

[Steedman2000] Mark Steedman. 2000. *The Syntactic Process*. Cambridge, MA, USA.

[Teh et al.2006] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

[van Zaanen2000] Menno van Zaanen. 2000. ABL: Alignment-based learning. In *Proceedings of the 18th International Conference on Computational Linguistics (Coling 2000)*, volume 2, pages 961–967, Saarbrücken, Germany.

[Varadarajan and Rao2009] Balakrishnan Varadarajan and Delip Rao. 2009. ϵ-extension hidden markov models and weighted transducers for machine transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 120–123, Suntec, Singapore, August.

[Vijay-Shanker and Weir1994] K. Vijay-Shanker and David Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

[Vogel et al.1996] Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th International Conference on Computational Linguistics*, volume 2, pages 836–841, Copenhagen, Denmark, August.

[Watkinson and Manandhar1999] Stephen Watkinson and Suresh Manandhar. 1999. Unsupervised lexical learning with categorial grammars using the LLL corpus. In *Proceedings of the 1st Workshop on Learning Language in Logic*.

[Witten and Bell1991] Ian H. Witten and Timothy C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

[Wong et al.2012] Sze-Meng Jojo Wong, Mark Dras, and Mark Johnson. 2012. Exploring adaptor grammars for native language identification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 699–709, Jeju Island, Korea, July.

[Wu1997] Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, September.

[Xiong et al.2010] Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Learning translation boundaries for phrase-based decoding. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 136–144, Los Angeles, California, June.

[Xue et al.2005] Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238, June.

[Yang et al.2009] Dong Yang, Paul Dixon, Yi-Cheng Pan, Tasuku Oonishi, Masanobu Nakamura, and Sadaoki Furui. 2009. Combining a two-step conditional random field model and a joint source channel model for machine transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 72–75, Suntec, Singapore, August.

[Zelenko2009] Dmitry Zelenko. 2009. Combining mdl transliteration training with discriminative modeling. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 116–119, Suntec, Singapore, August.

[Zhang et al.2010] Min Zhang, Xiangyu Duan, Vladimir Pervouchine, and Haizhou Li. 2010. Machine transliteration: Leveraging on third languages. In *Coling 2010: Posters*, pages 1444–1452, Beijing, China, August.

[Zhang et al.2011] Min Zhang, Xiangyu Duan, Ming Liu, Yunqing Xia, and Haizhou Li. 2011. Joint alignment and artificial data generation: An empirical study of pivot-based machine transliteration. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1207–1215, Chiang Mai, Thailand, November.

[Zhao and Gildea2010] Shaojun Zhao and Daniel Gildea. 2010. A fast fertility hidden markov model for word alignment using MCMC. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 596–605, Cambridge, MA, October.

[Zou and Hastie2005] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.