Treebank-Based Acquisition of Chinese LFG Resources for Parsing and Generation

Yuqing Guo

A dissertation submitted in fulfillment of the

requirements for the award of

Doctor of Philosophy

to the

Dublin City University

School of Computing

Supervisor: Josef van Genabith

July 2009

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Yuqing Guo

(Yuqing Guo)

Student ID: 55130887

Date: July 2009

Acknowledgements

The pursuit of a Ph.D. is a long journey full of obstacles and finally you will find no scenery is better than seeing white smoke rising from the exam room ... ☺

— adapted from the *The Odyssey*

There are many people who have helped get me to this point, to whom I am greatly indebted. First, I wish to express my deepest gratitude to my supervisor, Josef van Genabith, for his comprehensive guidance and supportive encouragement over the years, for his untiring enthusiasm for feeding me with new ideas, and for his incredible patience to correct and polish my broken English.

I gratefully acknowledge the valuable and in-depth discussions on the generation models with Aoife Cahill and Deirdre Hogan, which helped me to clear my mind, to shape my thinking and finally contributed to the generation chapters of this thesis. Many thanks to 方雾(Ji Fang) from PARC. It was such a pleasure to collaborate with her in developing the gold standard, even when we had different judgements on some linguistic issues. I extend my gratitude to those in this field who have ever given me commends, suggestions in academic activities and eventually had a great impact on this work. Special thanks to Tracy King, not only for the final feedback and advice on this thesis, but also for all the insightful comments and fruitful discussions at previous ParGram meetings and other conferences. Also thanks to my examiner Harold Somers for instructing me to cite references properly.

A collective thank you to Lamia, Jeniffer, Grzegorz, Natalie, Masanori, Yvette, Joachim and all the members of the NCLT lab. I have enjoyed being surrounded by

the varied, stimulating and relaxed atmosphere they provided in both the academic and social lives. And very big thanks to my neighbour Ines, who started and finished her Ph.D. at the same time as me, and who made those nights and weekends we spent together in the lab much less painful. I would like to thank all my friends in DCU, especially Krisztina, who is the first and best friend I made in Dublin. Furthermore, I am grateful to the instructors and teammates of the karate and badminton clubs, they have helped me to keep sane thoroughly and fit as a fiddle.

Exceptional thanks go to the Chinese community in DCU: 马艳军(Yanjun Ma), 孙艳丽(Yanli Sun), 葛某智(Mouzhi Ge), 焦德才(Decai Jiao), 刘璐(Lu Liu), 白亮(Liang Bai), 衡祥安(Xiang'an Heng), 杜金华(Jinhua Du), 郭华(Hua Guo), and all the others I could not name here, who made my life in Dublin more colourful and enjoyable.

I would also like to thank my former and current boss, 王海峰(Haifeng Wang) and the Toshiba fellows. Thank you for supporting me and offering me the opportunity to carry out work between DCU and Toshiba during the course of my Ph.D. study. And of course I am so lucky to rejoin the team in the end.

Most importantly, I am truly grateful to my family for their constant love, understanding and support throughout my education and life, without whom I would never have come this far.

Finally, thanks to the Science Foundation Ireland Grant 04/IN/I527 for the financial support of the GramLab project, enabling me to accomplish the research reported in this thesis.

Abstract

This thesis describes a treebank-based approach to automatically acquire robust, wide-coverage Lexical-Functional Grammar (LFG) resources for Chinese parsing and generation, which is part of a larger project on the rapid construction of deep, large-scale, constraint-based, multilingual grammatical resources.

I present an application-oriented LFG analysis for Chinese core linguistic phenomena and (in cooperation with PARC) develop a gold-standard dependency-bank of Chinese f-structures for evaluation. Based on the Penn Chinese Treebank, I design and implement two architectures for inducing Chinese LFG resources, one annotation-based and the other dependency conversion-based. I then apply the f-structure acquisition algorithm together with external, state-of-the-art parsers to parsing new text into "proto" f-structures. In order to convert "proto" f-structures into "proper" f-structures or deep dependencies, I present a novel Non-Local Dependency (NLD) recovery algorithm using subcategorisation frames and f-structure paths linking antecedents and traces in NLDs extracted from the automatically-built LFG f-structure treebank. Based on the grammars extracted from the f-structure-annotated treebank, I develop a PCFG-based chart generator and a new n-gram-based pure dependency generator to realise Chinese sentences from LFG f-structures.

The work reported in this thesis is the first effort to scale treebank-based, probabilistic Chinese LFG resources from proof-of-concept research to unrestricted, real text. Although this thesis concentrates on Chinese and LFG, many of the methodologies, e.g. the acquisition of predicate-argument structures, NLD resolution and the PCFG- and dependency n-gram-based generation models, are largely language and formalism independent and should generalise to diverse languages as well as to labelled bilexical dependency representations other than LFG.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Deep grammars relate strings to *meaning* representations in the form of logical forms, deep dependencies, or predicate-argument-adjunct structures. Most fundamental tasks in Natural Language Processing (NLP), such as Natural Language Understanding (NLU) and Natural Language Generation (NLG), would almost certainly benefit from deep and wide-coverage grammars. Traditionally, deep wide-coverage grammatical resources, particularly in unification or constraint-based formalisms such as Lexical-Functional Grammar (LFG) (Kaplan and Bresnan, 1982; Dalrymple, 2001), Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994), Combinatory Categorial Grammar (CCG) (Steedman, 1996) or Tree-Adjoining Grammar (TAG) (Joshi and Schabes, 1992) are developed by hand. Manual construction of such grammars is knowledge-intensive, time-consuming and prohibitively expensive, especially when:

- scaling beyond small fragments to unrestricted, real texts;
- migrating to diverse languages with typological differences.

In contrast to deep grammars, the relatively shallow syntactic formalism of Context-Free Grammar (CFG) is often adopted for building treebank resources (a large set of sentences hand-labelled with syntactic structures). In the last fifteen

years, the availability of treebanks such as the Penn Treebank (Marcus et al., 1993) has led to extensive efforts in treebank-based, statistical parsing models using machine learning techniques (Collins, 1999; Charniak, 2000; Klein and Manning, 2003; Petrov and Klein, 2007). These statistical parsers are trained automatically on the treebank without any requirement for a hand-crafted grammar, achieving broad-coverage, robustness and competitive performance. However, as many treebanks only encode purely syntactic information (and some functional roles and non-local dependencies by means of functional labels and empty nodes, which, however, are often ignored in the parsing models), the analysis provided by most of these parsers is too "shallow" to represent important semantic information, as compared to deeper unification or constraint-based grammars.

To overcome this dichotomy, in more recent years a growing body of research has emerged to automatically acquire TAGs (Xia, 1999; Chiang, 2000), CCGs (Hockenmaier and Steedman, 2002; Hockenmaier, 2003), HPSGs (Miyao et al., 2003, 2004) and LFGs (Cahill et al., 2002, 2008) from the Penn Treebank. These approaches combine the use of linguistically sophisticated, rich models of syntax and semantics with the data-driven methodology informed by probability theory and machine-learning techniques. A common characteristic of the research is that, to date, it has focused mainly on English. Chinese, one of the major languages in the world spoken by over 800 million people, and typologically very different from English, is currently the focus of much attention. Nevertheless, as yet natural language processing of Mandarin Chinese is far behind that of English: deep wide-coverage grammatical resources for Chinese are rare, and analysis of Chinese is almost entirely confined to shallow syntactic parsing or purely local dependencies.

The objective of the research described in this thesis is to produce treebank-based, wide-coverage and high-quality LFG resources for Chinese. In order to achieve these goals I investigate:

- the impact of language-specific properties of Chinese on the automatic acquisition of deep, wide-coverage, probabilistic, Chinese LFG resources from treebanks.

- the use and application of such automatically acquired resources to two fundamental NLP tasks for Chinese, namely, parsing and generation.

1.2 Background

1.2.1 Lexical Functional Grammar

The thesis is primarily grounded in the theory of LFG and investigates two fundamental issues in NLP, namely parsing and generation for Chinese, within the LFG framework.

Lexical Functional Grammar is a unification- or constraint-based grammar formalism introduced by Kaplan and Bresnan (1982) and further evolved in Kaplan (1995), Bresnan (2001) and Dalrymple (2001), etc. LFG posits two levels of syntactic representation that are most relevant to this thesis:

C(onstituent)-Structure uses a conventional CFG/phrase-structure trees encoding linear order, hierarchical groupings, and syntactic categories of constituents. It is used to represent language-particular constraints on word order and phrase structure.

F(unctional)-Structure characterises more abstract grammatical relations, such as SUBJ(ect), OBJ(ect), COMP(lement) or ADJUNCT. F-structures are hierarchical attribute-value matrices approximating to basic predicate-argument-adjunct structures or deep dependency relations. F-structures consist of the following types of attributes:

- PRED is the predicate/head of the local f-structure. The value of PRED is a *semantic form/subcategorisation frame* that is uniquely instantiated for each instance, reflecting the unique semantic contribution of each word within the sentence.
- Grammatical Functions (GFs) indicate the functional relationship between the predicate and dependents, whose values are subsidiary f-structures. GFs come in two different types:

  - governable GFs/arguments are subcategorised for by the local predicate, such as SUBJ(ect), OBJ(ect).
  - ungovernable GFs/modifiers like ADJUNCT, COORD(inate) are not subcategorised for by the predicate.

- Atomic-valued features describe linguistic (morphological or semantic) properties of the predicate by simple symbol values, such as TENSE, ASPECT, MOOD, PERS, NUM etc.

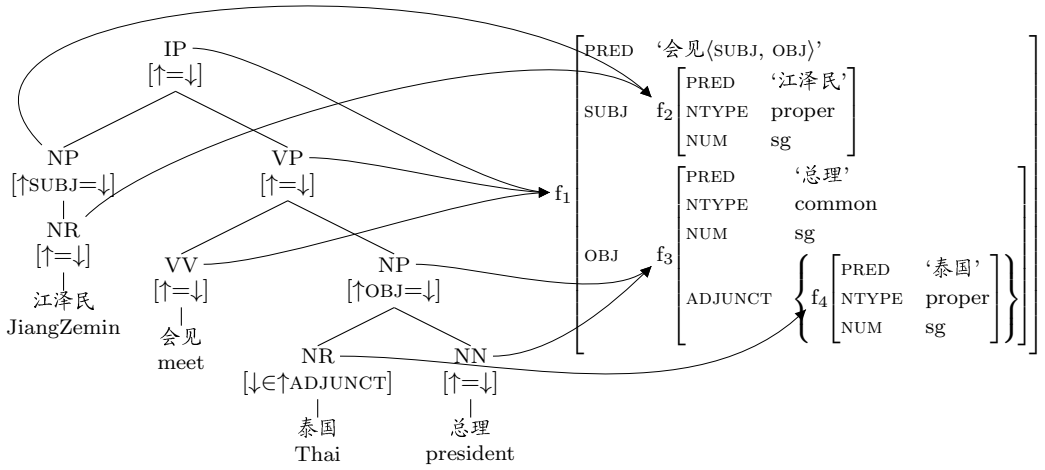
LFG imposes three general well-formedness conditions on f-structures licensing grammatically acceptable sentences (Kaplan and Bresnan, 1982):

Uniqueness: In a given f-structure a particular attribute may have at most one value.

Completeness: An f-structure is *locally complete* if and only if it contains all the governable grammatical functions that its predicate governs. An f-structure is *complete* if and only if it and all its subsidiary f-structures are locally complete.

Coherence: An f-structure is *locally coherent* if and only if all the governable grammatical functions that it contains are governed by a local predicate. An f-structure is *coherent* if and only if it and all its subsidiary f-structures are locally coherent.

Though LFG draws a sharp distinction between c-structures and f-structures, there are clear regularities relating constituent positions to grammatical functions. This structural correspondence is systematically expressed by a projection function ϕ mapping from nodes of the c-structure tree into units of the f-structure space (Figure 1.1). The ϕ function states universally valid relations between c-structure positions and the functional roles associated with them. To specify the ϕ function, nodes in c-structure trees are annotated with functional equations using language-particular principles. Functional equations employ two meta-variables: $\downarrow$ refers to the f-structure of the current c-structure node and $\uparrow$ refers to the f-structure of the immediately dominating node. For the example in Figure 1.1, the functional



(a) c-structure (b) f-structure

Figure 1.1: C- and f-structures for the sentence *江泽民会见泰国总理/JiangZemin met with Thai president*

equation ↑SUBJ=↓ annotating the left-most NP node indicates that the f-structure that corresponds to the node dominating the NP node (i.e. the IP node) has a SUBJ attribute whose value is the f-structure relating to the NP node.

This duality of syntactic structures in LFG is highly flexible to represent both variation and commonality in linguistic description. C-structures capture language-specific configurations, producing unique representations for different languages; f-structures encode a more abstract somewhat more universal level of analysis, supporting cross-language parallelism at this level of abstraction. These advantages facilitate cross-language applications, for example transfer-based machine translation systems which take f-structures as transfer representations, and motivate the work presented in this thesis.

1.2.2 Treebank-Based Acquisition of LFG Resources

1.2.2.1 The GramLab Project

The work reported in this thesis is part of the GramLab project that aims to automatically develop wide-coverage probabilistic Lexical Functional Grammar resources for Chinese, Japanese, Arabic, Spanish, French, German and English based on existing treebanks. Cahill et al. (2002) and McCarthy (2003) originally designed

an LFG f-structure annotation algorithm exploiting categorial, configurational and functional labels to automatically annotate CFG trees in the Wall Street Journal (WSJ) corpus of the Penn Treebank version II (Penn-II) with f-structure equations. Cahill et al. (2004) extended the method and used the f-structure-annotated treebank resources to automatically extract wide-coverage PCFGs for parsing new text into f-structures. It has been demonstrated that this methodology and the automatically acquired English LFG resources outperform state-of-the-art large hand-written grammars for the task of English parsing (Cahill et al., 2008).

This empirical success and the increasing availability of treebanks for other languages provide the motivation for adapting this approach to languages other than English. Burke et al. (2004) made an initial attempt to port the automatic f-structure annotation method to Chinese. Though the results reported in Burke et al. (2004) are promising, the f-structure annotation algorithm and the Chinese LFG resources developed are only proof-of-concept in that: (i) Chinese language-specific properties have not been thoroughly investigated and Chinese particular linguistic phenomena are not given fully appropriate treatments; (ii) raw texts are only parsed into basic and incomplete predicate-argument structures ("proto" f-structures) with non-local dependencies unresolved; and (iii) the experiments are only carried out on a relatively small-sized treebank — the Penn Chinese Treebank version 2 (CTB2, LDC2001T11) with about 4,000 sentences, which is less than one tenth of (about 50,000 sentences in) the WSJ section of the Penn-II English treebank. One of the most fundamental aspects of the research described in this thesis is a fundamental reworking and a substantial extension of the LFG acquisition method for Chinese language and treebank data, in order to provide high-quality, wide-coverage, deep, proper f-structure resources for Chinese language processing, in particular, parsing.

Another central aspect of the research in the thesis is surface realisation, or generation, from f-structures for Chinese. Natural language generation, although which is commonly regarded as the reverse process of parsing, has drawn a lot less attention in the field of NLP, and has not been attempted for languages other than English in the GramLab project. Cahill and van Genabith (2006) and Hogan et al. (2007)

presented conditional probabilistic models for generating English sentences from f-structure representations based on the automatically acquired wide-coverage LFG resources. The generation part of this thesis is based on and substantially extends this work: the pros and cons of the conditional generation model are investigated when adapting it to the Chinese generation task and a novel, fully PCFG-based generative model is developed.

1.2.2.2 Language-Specific Properties of Chinese

Among the languages under consideration in the GramLab project, Chinese is one of the most challenging languages. Besides the obvious differences, such as the writing systems, Chinese grammar features very distinct characteristics compared to Indo-European languages.

Compared with even English, Chinese has little, if any, inflectional morphology, viz. words are not inflected with respect to tense, case, person, gender and number, etc. As a result, there are no overt syntactic constraints, such as agreement, imposed between a verbal predicate and its arguments. Moreover, every word in Chinese has a unique form regardless of its potentially varied syntactic functions. For example about 80% of the most common verbs in Chinese can also function as a noun and this without carrying any conjugation or declension, which leads to a large number of ambiguities in part-of-speech tagging and syntactic parsing.

Besides the lack of morphological marking, cues such as complementisers or case markers, which can be used to distinguish between syntactic analyses, are very rare in Chinese. For example, the matrix verb *要求/ask* in (1) can be equivalently analysed as either taking an object and an open complement as arguments (a) or taking a close complement (b):

(1) 我 要求 他 过来。
I ask he come

a. I asked him to come.
b. I asked that he come.

Since words are not marked morphologically indicating their roles in a sentence, function words and word order play major roles in Chinese. There are a consider-

able number of specific constructions governed by particular words, such as 的/*de*, 地/*di*, 得/*de*, 把/*bai*, 被/*bei* and so on. These constructions are vital in analysing the structure of the sentence in which they occur, and thus need a careful and appropriate treatment.

Two further prominent features of Chinese are *pronoun-dropping* and *topic-prominence*. The phenomenon of *pro-drop* is widespread in Chinese as e.g. the subject (and likewise the object) is only semantically but not syntactically required for constructing a grammatical sentence. Pronouns occurring in a subject (or object) position are often omitted if they are inferable from the context. Topicalisation is also fairly common in Chinese. The topic takes sentence-initial position in a topicalised sentence, whereas a nontopicalised sentence follows the unmarked subject-first word order. Topics in Chinese can bear a variety of functions including both arguments and adjuncts. The two properties (pro-drop and topic-prominence) together (among other syntactic features) cause difficulties in both (i) identifying pro-drop situations and recovering the pro-dropped subjects; and (ii) distinguishing topics from subjects (and other adjuncts) and relating the topic to the appropriate grammatical function it should be interpreted as.

As a compensation for almost complete lack of morphological information, word order tends to carry a lot of information in Chinese. For example, Chinese is by and large a SVO language in which grammatical functions are to a large extent related to a relatively rigid word order. The rather direct mapping between grammatical roles and surface realisations drives a new approach to Chinese sentence generation obviating constituent structure grammar rules presented in Chapter 6.

In summary, Chinese is a language with poor morphological features and consequently few agreement constraints, relying instead more heavily on lexical items and word order. Although the thesis is not aiming at establishing a definitive and fully theoretically adequate LFG analysis for Chinese, it carefully considers the language-specific properties of Chinese and provides feasible accounts for Chinese syntactic and semantic interpretation within the LFG formalism from a computational perspective.

1.2.2.3 The Penn Chinese Treebank

The treebank on which my research is based is the Penn Chinese Treebank (CTB) (Xue et al., 2005). The Chinese Treebank project began at the University of Pennsylvania in 1998 and continues at Penn and the University of Colorado. The data I use comes from the Chinese Treebank version 5.1 (CTB5.1) produced and distributed by the Linguistic Data Consortium (catalog number LDC2005T01, ISBN 1-58563-323-2). The CTB5.1 is a corpus of Mandarin Chinese texts consisting of 507,222 words (824,983 Chinese characters), 18,804 sentences, and 890 files coming from three newswire sources:

- Xinhua newswire (1994-1998): 698 articles;
- Information Services Department of HKSAR (1997): 55 articles;
- Sinorama magazine, Taiwan (1996-1998 & 2000-2001): 137 articles.

Texts in CTB are segmented, part-of-speech tagged (with 33 tags), and syntactically bracketed (with 23 labels) based on a generic phrase structure grammar analysis. The influence of the English Penn Treebank on the development of the Chinese Treebank is obvious. The CTB essentially adopts the Penn Treebank annotation scheme and uses the same grammatical devices to represent syntactic relations:

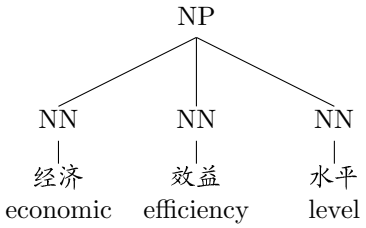
- a limited number of grammatical relations, such as subject, direct and indirect object, etc. are represented by 26 functional tags;
- 6 types of null elements and coindexation to represent non-local dependencies.[1]

Similar to the Penn-II treebank, overflat bracketing is a rather common phenomenon for representing nominal constructions in the CTB, for instance no further bracketing is provided inside the noun phrase (2).

Nevertheless, the CTB provides a large-scale corpus with useful and in many cases fine-grained linguistic information. In fact, the CTB has become the de facto standard training and test set for many NLP tasks, such as statistic parsing for Chinese.

[1]The CTB bracketing guidelines gives the full inventory of null elements and functional tags.

(2) 经济 效益 水平
economic efficiency level
'level of the economic efficiency'



1.3 Thesis Outline

This thesis consists of two parts focusing on two complementary NLP tasks for Chinese, both involving LFG f-structures. The first part consists of Chapters 2 and 3, and presents the core resources for parsing Chinese into f-structures. The second part consists of Chapters 4 to 6, addresses the issue of generation from f-structures for Chinese. An overview of the thesis is given below:

Chapter 2 presents two treebank-based architectures for automatically constructing Chinese LFG resources from the CTB that will be used throughout the thesis. First, I describe an extensive overhaul, further development and substantial extension of the preliminary f-structure annotation algorithm of Burke et al. (2004). Second, I provide an alternative conversion-based approach to transforming treebank trees into f-structures directly. Finally, I report experiments with the both methods to acquire f-structures from treebank trees and on parsing unseen text into f-structures.

Chapter 3 presents strategies to improve the quality of the f-structure resources induced for raw text. The f-structures generated from c-structure trees produced by state-of-the-art parsers are "proto" f-structures, which do not capture non-local dependencies (NLDs). I design and implement a hybrid NLD resolution method inspired by Cahill et al. (2004), but substantially extended and adapted to Chinese-specific NLD phenomena. The approach automatically resolves NLDs at the level of f-structure by a combination of heuristics and a statistical component using finite

approximations of functional uncertainty equations and lexical subcategorisation frames automatically acquired from the f-structure treebank constructed in Chapter 2.

Chapter 4 provides an overview of the state-of-the-art in natural language generation, in particular sentence realisation, and proceeds to give a brief introduction to previous research on LFG-based generation. It also discusses problems specific to Chinese sentence realisation from f-structures, which motivate a direct generation methodology driven by functional relations instead of the detour via syntactic structures.

Chapter 5 describes the adaptation of a PCFG- & chart-based generation method from English to Chinese, using the bidirectional grammar automatically acquired from the f-structure annotated treebank. Inspired by the conditional probabilistic model proposed by Cahill and van Genabith (2006), I design a proper generative and chart-based PCFG generation model, which effectively increases the generation coverage compared to the original conditional model. I investigate two parent annotation methods to break down inappropriate independence assumptions inherent in the vanilla PCFG by including more contextual information into the derived tree. The augmented PCFG models further enhance the accuracy of the simple PCFG model for Chinese sentence generation.

Chapter 6 describes a novel approach to Chinese sentence realisation based on dependency (rather than word-based) n-gram models, which directly linearise the GFs in f-structures without recourse to an underlying CFG grammar and chart mechanism. Experiments show that the dependency-based n-gram models are superior to traditional PCFG-based generation models in terms of time complexity and realisation quality.

Chapter 7 concludes the thesis and outlines some avenues and applications for future related research.

Part I

LFG-Based Parsing

Chapter 2

Automatic F-Structure Acquisition

2.1 Introduction

Deep, board-coverage and high-quality grammars are required in many NLP applications, such as deep parsing and generation from semantic representations. In this chapter, I report work on automatically acquiring wide-coverage, robust, probabilistic Chinese LFG resources from the Penn Chinese Treebank, which forms the basis of the approaches to Chinese parsing and sentence realisation presented in the following chapters.

One goal of the thesis is to test a treebank-based deep grammar acquisition methodology originally developed for English (Cahill et al., 2002, 2004) for its applicability when migrating it to diverse languages and treebanks. I review and substantially extend and improve the early and preliminary work of Burke et al. (2004) by improving the LFG analysis for a number of core Chinese linguistic phenomena and expanding the coverage of the treebank-based grammars. This results in Chinese LFG resources comparable in quality to those for English. Furthermore, I address some inherent drawbacks in the previous annotation-based acquisition method, and develop an alternative conversion-based architecture.

Section 2.2 reviews previous research on LFG grammar development. Section 2.3

reports the work on improving the earlier, proof-of-concept work on automatically annotating the CTB with LFG f-structure information (Burke et al., 2004). Section 2.4 presents a new conversion-based architecture which involves an intermediate dependency representation to further enhance the robustness of the annotation algorithm and increase the coverage of the grammar induced. Section 2.5 provides experiments and evaluates the two approaches to f-structure acquisition. Section 2.6 applies the automatic f-structure acquisition technology to "shallow" syntactic trees produced by existing, state-of-the-art parsers to parse raw texts into proto-f-structures or dependencies. Parts of the research reported in this chapter have been published in Guo et al. (2007a).

2.2 Previous Work on LFG Grammar Development

LFG resources are usually developed by hand. The ParGram[1] project aims at producing large-scale LFG grammars for a wide variety of languages with largely parallel analyses, involving commonly agreed feature sets and where possible f-structure analyses across languages. The results of the project to date are encouraging: the hand-crafted English LFG grammar has in fact achieved the coverage and robustness required to parse a corpus of the size and complexity comparable to the Penn treebank (Riezler et al., 2002), and grammars for distinct languages such as Japanese, Chinese, German, Urdu etc. are under development and some of them (German, Japanese) are very mature. The ParGram project started in 1994 and involves substantial number of researchers and linguists in industrial and academic institutions around the world, defining the commonly-agreed-upon set of features, developing grammar and lexicon rules and maintaining parallelism across languages (Butt et al., 1999, 2002).

However, the availability of treebank resources opens up the possibility of automatically acquiring deep, wide-coverage grammars much more rapidly compared to manually creating such resources. Over the last decade, a growing body of research (van Genabith et al., 1999; Sadler et al., 2000; Frank, 2000; Cahill et al.,

[1] http://www2.parc.com/isl/groups/nltt/pargram/

2002) has emerged to bootstrap the construction of LFG grammars from f-structure corpora automatically built by annotating existing treebank (c-structure) trees with f-structure information.

Annotation-Based Approaches

LFG has shown that the correspondence between c-structure and f-structure is largely predictable from a small set of mapping principles (Bresnan, 2001). In Kaplan and Bresnan (1982), the c- to f-structure mapping is represented in terms of constraints on pairings of categories and grammatical functions, e.g. COMP is usually only appropriate for S, only NPs or DPs are OBJs, and so forth. In general, the correspondence between c-structure and f-structure follows linguistically determined principles and can be captured in terms of general annotation patterns.

In the light of this observation, an "annotation"-based LFG grammar acquisition approach has been developed. van Genabith et al. (1999) automatically extract CFG rules from a treebank and manually annotate the derived grammar rules with functional schemata and lexical macros. Then the annotated rules are matched against the original treebank trees and thereby f-structures are produced for these trees. This method supports the creation of grammar resources but still involves a labour intensive component to manually annotate the grammar rules extracted from the treebank. To automate the manual part, Sadler et al. (2000) developed a smaller number of hand-crafted templates using regular expressions encoding general annotation principles, and applied these principles to automatically annotate the treebank-extracted CFG rules.

However, both papers are proof-of-concept in that they only experiment on grammars consisting of a few hundred rules extracted from the first 100 trees of the Associated Press (AP) treebank. Cahill et al. (2002) extended and scaled up an annotation-based method to the complete WSJ section of the Penn-II treebank. The approach builds f-structure resources in a two-stage process (Figure 2.1): first, the treebank trees are annotated with functional equations by an f-structure annotation algorithm, and then the f-structure-annotated trees are passed to a constraint solver

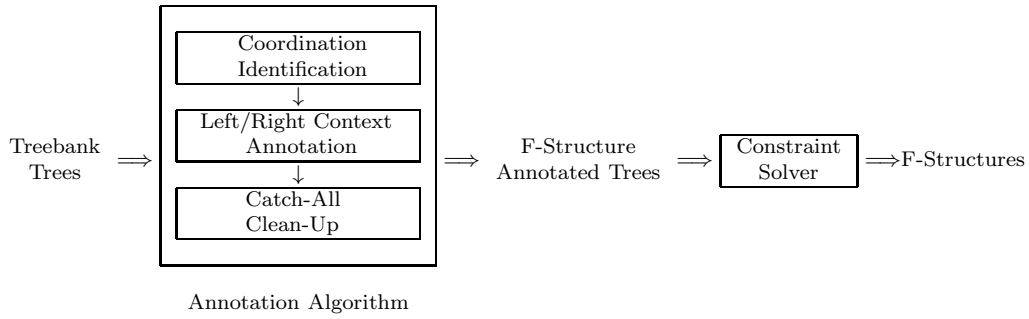


Figure 2.1: The f-structure annotation architecture of Cahill et al. (2002)

to generate f-structures. The annotation algorithm consists of three sub-modules that work in sequence. Left/right context annotation is the core module which applies to each local subtree of depth one (i.e. a CFG rule). First, the RHS nodes are partitioned into head, left context and right context of the head by head-finding rules designed for the treebank. The head node is annotated with f-structure equation $\uparrow=\downarrow$, and each node in the left/right context is annotated accordingly by annotation principles based on categorial and configuration information. For example, an NP node occurring to the right of a V head under a VP is annotated as the object $\uparrow$OBJ$=\downarrow$. In order to keep the left-right context principles simple and perspicuous, they only apply to non-coordinate structures. A separate module is designed to deal with coordinations. It annotates the head coordinating conjunction, and identifies all coordinated elements which in turn are annotated by regular left/right context annotation principles. The final module (Catch-All and Clean-Up) corrects over-generalisations that arise from the previous annotation modules and assigns default annotations to any remaining unannotated nodes.

Compared to assigning functional schemata or designing templates as regular expressions for all CFG rules, the annotation scheme of Cahill et al. (2002) provides annotation matrices by breaking down CFG rules into left/right contexts. The left/right context matrices are based on the most frequently-occurring rule types (covering 85% rule tokens of Penn-II for each category), but provide generalisations over the entire treebank and as yet unseen rules. Therefore annotating c-structure tree nodes with f-structure equations can be implemented in a highly general way

and scaled to a large-size corpus. Cahill et al. (2002) report 78.8% of the 49K Penn-II sentences receiving a single, connected f-structure, which is later improved to 99.82% in Cahill (2004).

The treebank and annotation-based f-structure acquisition methodology is originally developed for English and later also deployed for multilingual grammar acquisition, given treebank resources for those languages are available. Burke et al. (2004), for example, made a first attempt to adapt the technology of Cahill et al. (2002) to the Penn Chinese Treebank for Mandarin Chinese. I report on substantial extensions, revisions and improvements over this previous work in Section 2.3.

Conversion-Based Approaches

Another related but different architecture for f-structure acquisition from treebanks is referred to as "conversion"-based, which directly induces an f-structure from a c-structure tree, without intermediate functional schemata annotations on the c-structure tree. An algorithm building on this architecture was developed in Frank (2000), where (complex) c-structure fragments encoded as constraint sets were converted directly into corresponding f-structure components using a term rewriting system originally developed for transfer-based machine translation. As opposed to the CFG rule- and annotation-based architecture in which annotation principles are by and large restricted to local trees of depth one, this approach is naturally applicable to non-local tree fragments.

From a computational perspective, LFG f-structure is effectively a dependency format. In this sense, the task of generating f-structures from c-structures is essentially identical to converting Phrase Structure (PS) trees to Dependency Structure (DS) trees. There has been a lot of linguistic discussion on the comparison between dependency grammars and context-free phrase structure grammars, and some research (Lin, 1995; de Marneffe et al., 2006; Xue, 2007) on automatic conversion between PS and DS representations. Tools such as Penn2Malt[2] have also become available for dependency parsing and dependency-based parser evaluation. Trans-

[2]The tool is downloadable at http://w3.msi.vxu.se/ nivre/research/Penn2Malt.html

formation algorithms presented in these papers, in the main, use head percolation rules to determine the head node of each local PS tree and recursively "fold" the PS tree into a corresponding DS tree by percolating the head nodes.

In Section 2.4, I explore this more general conversion-based approach in combination with the left/right context annotation principles to construct LFG f-structures from treebank phrase structure trees.

2.3 Annotating Treebank Trees

2.3.1 Comparison to Previous Work

An initial attempt to port the treebank- and annotation-based LFG acquisition methodology to Chinese data was carried out by Burke et al. (2004), which applied a generic version of the annotation algorithm of (Cahill et al., 2002; Cahill, 2004) adapted to the Penn Chinese treebank. The experiments were proof-of-concept and limited with respect to (i) the size of the treebank and gold standard tested against; (ii) the soundness of the LFG analysis of some Chinese linguistic phenomena; (iii) the coverage and accuracy of the annotation algorithm; (iv) only producing "proto" f-structures for parser output trees with NLDs unresolved.

Addressing these shortcomings, my research substantially extends and improves on the previous work in the following aspects:

- The annotation algorithm is scaled up to the full Penn Chinese Treebank version 5.1 (CTB5.1), whose size is more than 4 times of that of the earlier CTB2 used in Burke et al. (2004).

- A more deliberate and extensive analysis is made for the LFG f-structure representation of Chinese linguistic phenomena, considering both Chinese-specific properties and cross-linguistic parallelism. Some f-structure analyses different from the previous work are exemplified in Section 2.3.2.

- According to the revised LFG analysis, the earlier set of 50 gold-standard f-structures are modified, and in addition a new extended set of Chinese gold-

standard f-structures for 200 sentences is developed (jointly with PARC) for evaluation.

- A postprocessing module is developed for resolving Chinese NLDs in the proto-f-structures produced by the annotation algorithm for parser output trees, and proper-f-structures are generated. The approach will be detailed in Chapter 3.

- Annotation principles of the core algorithm are thoroughly examined and substantially extended.

To seed the generic annotation algorithm with linguistic generalisations for Mandarin Chinese, Burke et al. (2004) extracted the most frequent CFG rule types for each phrasal category in the CTB2 with joint coverage of $\geq 85\%$ of total rule token occurrences for that category. These seed rule types were then annotated with f-structure equations. The set of fully annotated seed rules played an important role in the previous annotation algorithm for Chinese: (i) they were used to construct a preliminary version of left/right annotation matrices consisting of about 100 rules only, which was far from providing fully-fledged and fine-grained annotation, hence (ii) the completely annotated rules were kept and used directly in the annotation process. Burke et al. (2004) reported 96.75% of the trees in CTB2 receiving a single, covering and connected f-structure by the annotation algorithm. However the figure dropped to 88.17% when the algorithm was applied to the larger-sized CTB5.1. To increase coverage, I enlarged the set of left/right annotation rules three times to a total of 300 rules. As most CFG seed rules are already covered by the left/right context patterns, they do not take part in annotating the treebank trees any longer.[3] Moreover, in Burke's version of the annotation algorithm, coordination constructions were also tackled by completely annotated seed rules, which did not generalise well due to their particularities and failed to identify many types of coordination structures in the CTB. I substantially improved this by implementing a separate coordination module with more general rules or patterns, as proposed in

[3]Some very few CFG annotated rules that cannot be expressed in left/rigth context patterns, are still kept for annotating particular constructions in the CTB, such as overflat coordinations and fragments.

the annotation architecture for English (Cahill et al., 2002).

The new annotation algorithm with improved annotation principles and modules gives considerably better results than the previous work in terms of both coverage and accuracy. An overall evaluation is reported in Section 2.5.

2.3.2 Improving and Extending the LFG Analysis for Chinese

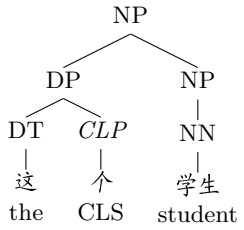
Work addressing Chinese within the LFG formalism has been carried out only for a limited number of phenomena. For example, Fang (2006) provided a formal analysis for the verb copy construction in Chinese; Huang and Mangione (1985) offered an LFG account of the post-verbal 得/DE-construction; Her (1991) presented a classification of Mandarin verbs by subcategorised grammatical functions within LFG. However, to the best of my knowledge, there has not yet been a standard and systematical LFG account for Chinese. On the contrary, there have been a lot of controversies over syntactic analysis of Chinese from the theoretical linguistics perspective, and, from a practical point of view, the f-structure representations for some Chinese linguistic phenomena adopted by Burke et al. (2004) do not suit the LFG formalism nor the encoding of the Penn Chinese Treebank well. To give a flavour of the improvements of the LFG-based analysis for Chinese, this section exemplifies c-structure trees provided by the CTB and my enhancements to the corresponding f-structure analysis for core Chinese linguistic constructions.

Classifiers[4] are unique to Chinese (and some other Asian languages) in that all nouns, when occurring with a numeral or demonstrative, generally incorporate a classifier, rather than in most Indo-European languages, allowing numbers to count a noun directly. By and large, classifiers occur in three types of context, based on the absence/presence of the numeral and demonstrative. Figure 2.2 gives the CTB trees for classifiers used in combination with a demonstrative (1), a numeral (2) and both (3).

In the previous analysis of Burke et al. (2004), a classifier was analysed as a

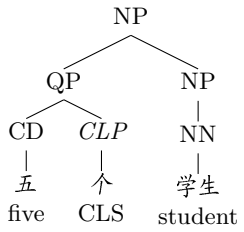
[4]Also called "measure words", for they are most often used when counting.

(1) 这 个 学生
this *CLS* student
'this student'



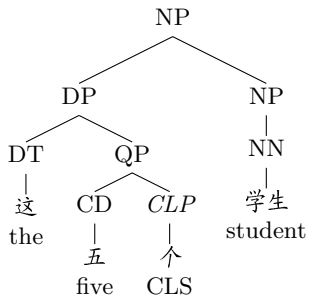
$$\begin{bmatrix} \text{PRED} & \text{'学生'} \\ \text{DET} & \begin{bmatrix} \text{PRED} & \text{'这'} \end{bmatrix} \\ \text{QUANT} & \begin{bmatrix} \text{PRED} & \text{'个'} \\ \text{NUMBER} & \begin{bmatrix} \text{PRED} & \text{'一'} \\ \text{PRED-FORM} & \text{null} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(2) 五 个 学生
five *CLS* student
'five students'



$$\begin{bmatrix} \text{PRED} & \text{'学生'} \\ \text{QUANT} & \begin{bmatrix} \text{PRED} & \text{'个'} \\ \text{NUMBER} & \begin{bmatrix} \text{PRED} & \text{'五'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(3) 这 五 个 学生
these five *CLS* student
'these five students'



$$\begin{bmatrix} \text{PRED} & \text{'学生'} \\ \text{DET} & \begin{bmatrix} \text{PRED} & \text{'这'} \end{bmatrix} \\ \text{QUANT} & \begin{bmatrix} \text{PRED} & \text{'个'} \\ \text{NUMBER} & \begin{bmatrix} \text{PRED} & \text{'五'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 2.2: The CTB trees and my f-structure analysis of classifiers

COMP dependent on the numeral if it exists or the demonstrative in case the numeral is absent. This analysis is defective, in that:

- Though classifiers cannot be used by themselves and must be preceded by a numeral or demonstrative, it does not follow that numerals and demonstrative pronouns subcategorise for a classifier in Chinese or other languages. In fact, classifiers do not need to be expressed in some cases, having no effect on the

grammaticality, such as:

(4) 这 书 真 有趣
this book very interesting
'This book is very interesting.'

- The analysis is not suitable for standard classifiers, such as *米/meter*, *公斤/kilogram*, *瓶/bottle*, which relate to distance, weight, volume, etc. These are analogous to measure words representing units or portions of mass nouns in other languages, such as English, for example 'one *drop* of milk', where the measure word functions as the main predicate.
- Individual classifiers also indicate prominent features of the noun they modify, e.g. *把/BA* is derived from *handle* and is usually used as a classifier for objects with a handle, for instance, a chair. As a consequence, head nouns can be omitted in some cases where a classifier is then used as a replacement for it, as in (5). This type of relationship between the classifier and the noun it modifies is not revealed in Burke's analysis.

(5) 一 本 多少 钱
one *CLS* how much money
'How much is for one (book)?'

Though as indicated by the name, classifiers act to classify nouns according to their meaning, they actually are most often used when counting in combination with a numeral. Even if a classifier is used alongside a demonstrative as in example (1), it can be interpreted as an omission of the unstressed numeral *一/one*. Therefore, I provide a unified and somewhat more practical interpretation of classifiers, as demonstrated by the f-structures in Figure 2.2 for the three types of c-structure trees. In my analysis, a numeral is dependent on the classifier (which accounts for their concurrence), and they together quantify the head noun. A classifier is never associated with a determiner in my f-structure analysis (even if it is in the CTB trees), reflecting the fact that DET and QUANT are parallel functions specifying the PRED noun in different ways.

DE-Phrases are formed by the function word 的/*DE* attached to various categories, such as possessive phrases, noun phrases, adjective phrases or relative clauses. 的 has no content other than marking the preceding phrase as a modifier. In the original f-structure annotation algorithm and the 50-sentence gold-standard f-structures of Burke et al. (2004), all types of DE-phrases were treated uniformly in the corresponding f-structures: the function word 的 is the predicate that dominates the local f-structure and the preceding phrase/clause is annotated as possessive. This analysis is inadequate for the following reasons:

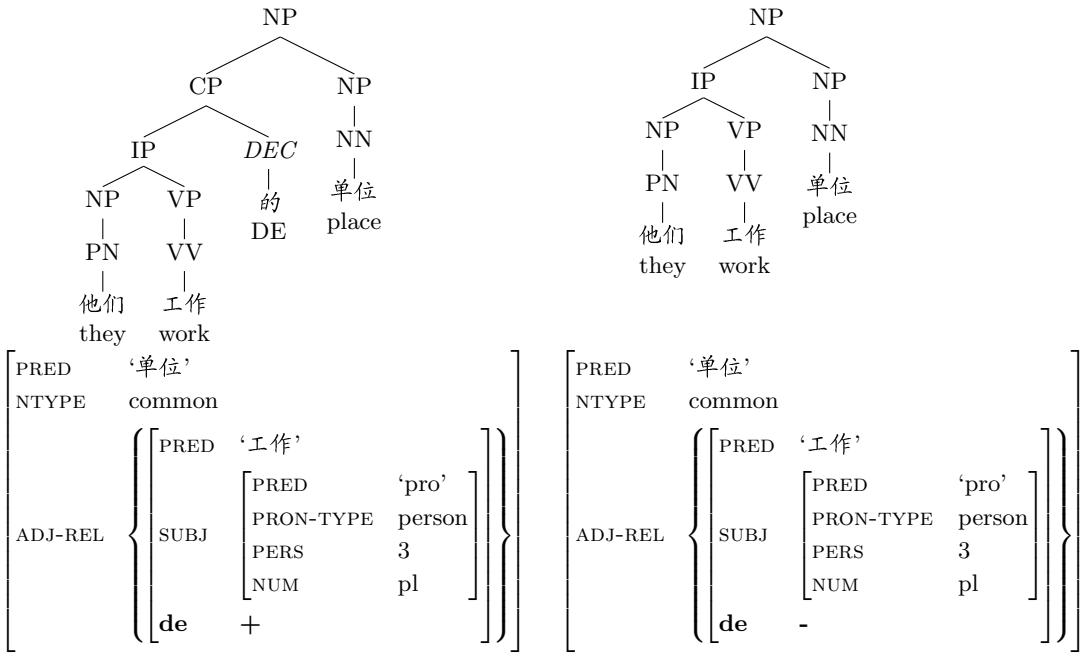
- 的 is the syntactic head in DE-Phrases, but not the semantic head as the real meaning is carried by the phrase it attaches;
- 的 has no content, therefore it can be omitted in some cases without affecting the meaning and grammaticality of the phrase, as in (6a) and (6b).[5] In this regard, there is some resemblance between the function word 的 and relative pronouns/adverbs in English relative clauses;
- In many cases, 的 is a mark of an attributive modifier, qualifying the head noun by a variety of properties besides its possessor.

Bearing in mind that f-structure is an abstract functional description of a sentence, and thus preferably less language- and treebank-specific, I choose the content word rather than 的 as main predicate of the local f-structure. In parallel to f-structures for English relative clauses, I treat 的 as an optional feature *de* of the modifier f-structure. Moreover, my analysis distinguishes three different types of DE-modifiers in the f-structures as exemplified in Figure 2.3, namely ADJ-REL (6), ADJUNCT (7) and POSS (8). Due to the absence of any case marking, Chinese uses the function word 的 to indicate possessive function as in example (8), which however shares the same local c-structure tree (*NP→DNP NP*) as a normal attributive modifier as in example (7). The difference in fact resides in the meaning of the lexemes, that is, the head word of the ADJUNCT is a common noun (NN), and the head word of the POSS is a proper noun (NR).

[5]For simplicity, empty nodes and co-indexation representing non-local dependencies are left out in both c- & f-structures here.

(6) a. 他们 工作 的 单位
they work *DE* place
'the place where they work'

b. 他们 工作 单位
they work place
'the place they work'



(7) 大 规模 的 项目
large scale *DE* project
'a large-scale project'

(8) 张三 的 书
ZhangSan *DE* book
'ZhangSan's book'

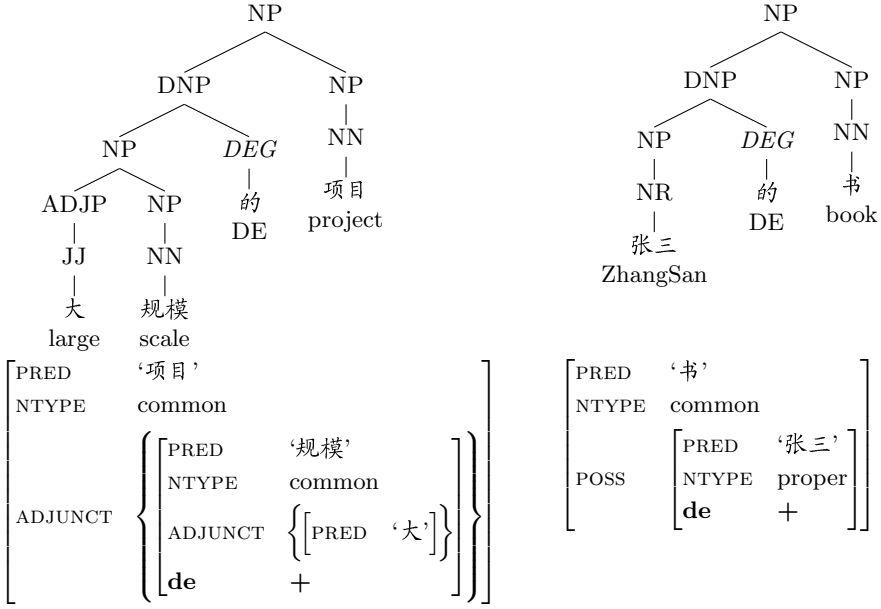
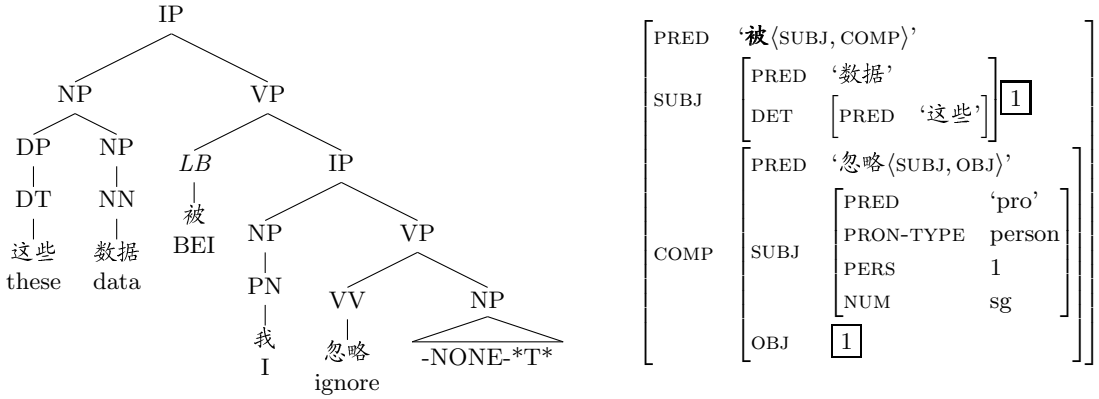


Figure 2.3: The CTB trees and my f-structure analysis of DE-phrases

(9) 这些 数据 被 我忽略
these data *BEI* I ignore
'These data were ignored by me.'



(10) 他 被 授予 一等奖
he *BEI* award the top prize
'He was awarded the top prize.'

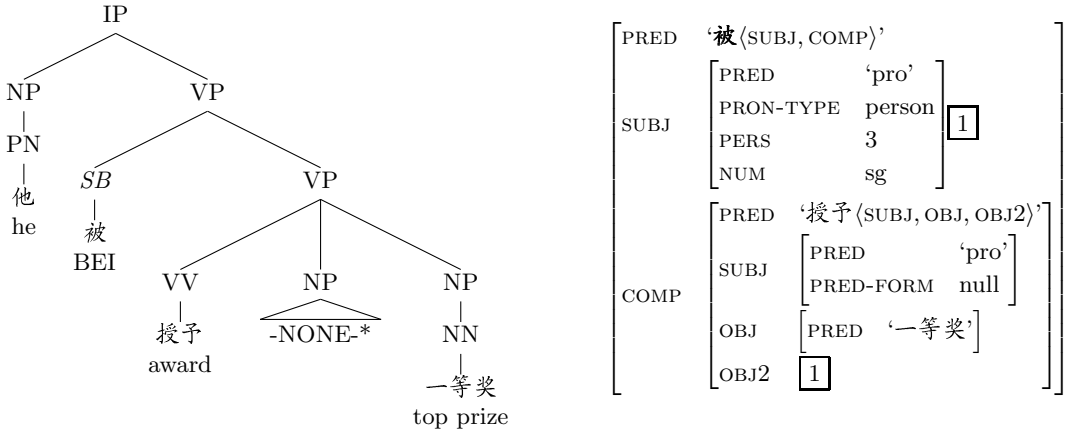


Figure 2.4: The CTB trees and my f-structure analysis of BEI-constructions

BEI-Construction is another characteristic phenomenon in Chinese, but did not receive special treatment in Burke's annotation algorithm, other than simply annotating the word *被/BEI* as an ADJUNCT. Generally speaking, the BEI construction is approximately equivalent to passive voice in English. A proposal analogous to the representation of English passive voice is to take *被* as a passive voice feature of common verbs. However, treating *被* as a mere passive feature or particle contradicts the fact that it also introduces the logical subject in the long-BEI construction,

as illustrated in (9). In this case, the word 被 is more similar to the preposition *by* in the equivalent English passive translation. Unfortunately, analysing 被 as a preposition or subject marker does not fit the subjectless/short-BEI construction as in (10), where 被 introduces or marks nothing. And furthermore, 被 does not always indicate passive voice in Mandarin Chinese (though the passive analysis can be argued for from a theoretical point of view), as in sentence (11), where the embedded verb is intransitive.

(11) 猫 被 老鼠 跑 了
cat *BEI* mouse escape ASP
'The cat let the mouse escape.'

Recent years have seen a radically different analysis that treats 被 as a content word rather than a function word as in traditional Chinese linguistics, more specifically as a verb with its own predicate-argument structure. Her (1991) provided a complete and unified analysis for 被 following this line, which treats 被 as the matrix verb in a pivotal construction, subcategorising for $\langle \text{SUBJ}, \text{OBJ}, \text{XCOMP} \rangle$. The advantage of this analysis is that it provides a unified account for the embedded verbs, namely that all verbs in BEI sentences have the same subcategorisation frames as those in their BEI-less corresponding sentences. However, this is somewhat different from the CTB representation, where 被 takes either a subject and a sentential complement in the long-BEI construction (9), or a subject and a VP complement in the short-BEI construction (10). As there is not always a clear distinction between finite clauses and non-finite clauses in Chinese due to the lack of overt complementisers and a strong tendency to pro-drop, I feel both Her's and CTB's analyses are acceptable. For practical reasons, I adopt the tree representation in CTB as the c-structure analysis for BEI constructions, and give the corresponding f-structure analysis as illustrated in Figure 2.4. Though c-structures for short-BEI and long-BEI constructions vary from each other, I provide a unified f-structure account for 被 with the subcategorisation frame $\langle \text{SUBJ}, \text{COMP} \rangle$, and supply a pro-dropped subject to the verb embedded in the short-BEI construction.

2.4 Converting Treebank Trees

The f-structure annotation method described in the previous section builds on the CFG rule- and annotation-based architecture. By and large the algorithm works on local treebank subtrees of depth one (equivalent to a CFG rule). In order to annotate the nodes in the treebank tree with f-structure functional equations, the algorithm exploits configurational information (left or right position relative to the head), category of mother and daughter nodes, and Penn treebank functional labels (if they exist) on daughter nodes. However configurational and categorial information from local trees of depth one is not always sufficient to determine the appropriate functional relations between the parent and daughter nodes, as for the example of DE-phrases, where a left-context DNP node under NP node can be annotated as either ADJUNCT (7) or POSS (8) in Figure 2.3. Furthermore, due to the level of generalisation of the left/right context annotation principles, inaccurate functional equations might be assigned during the annotation and should be corrected by the clean-up module. For example, any NP nodes occurring to the left of the head under an IP node are assigned ↑SUBJ=↓ (if no CTB functional labels are available to support a different analysis). This has the unfortunate consequence that in Figure 2.5, the three NP nodes under an IP are all assigned with ↑SUBJ=↓ (whereas in reality n_1 functions as a modifier, n_2 as topic and n_3 as subject, respectively). Annotating more than one SUBJ for the same PRED would result in an irresolvable set of equations preventing the generation of an f-structure. In this example, it is easy to discover the feature clash between the NP nodes n_1 and n_2, as they are in the same local subtree dominated by the root IP node. However it is much more difficult to find the conflicting functional equation annotated on n_3 since it is located in a lower level subtree. To tackle these problems, access to lexical information and wider contextual information beyond the local configurational and categorial structure are required by annotation principles.

The annotation-based f-structure acquisition approach is also compromised by limitations of the constraint solver used to generate f-structures from the set of functional equations collected from the annotated treebank trees. As explained in

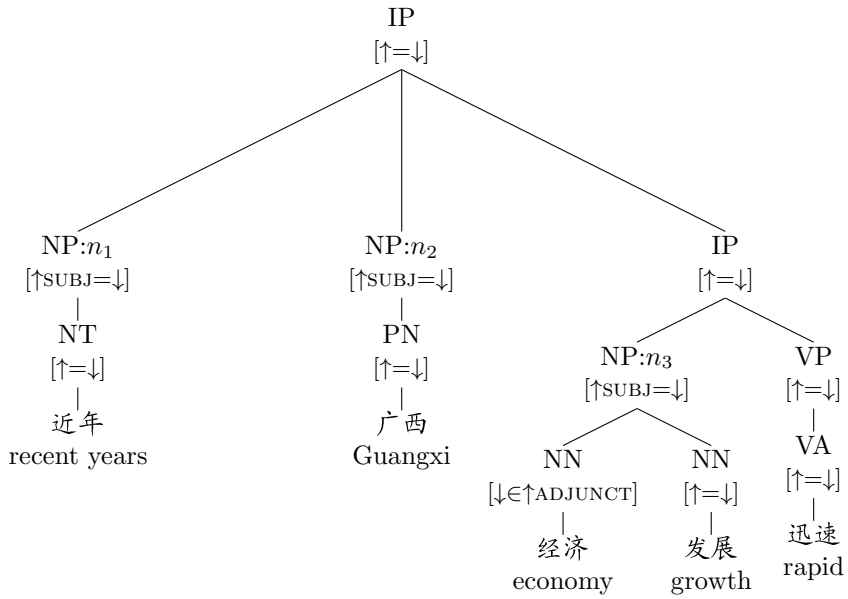


Figure 2.5: General left/right context annotation on the treebank tree for the sentence *近年广西经济发展迅速/The economy of Guangxi province has grown rapidly in recent years*

Cahill et al. (2002), the constraint solver can handle equality constraints, disjunction and simple set-valued feature constraints. However, it (i) fails to generate an f-structure (either complete or partial) in case of clashes between the automatically annotated features; and (ii) does not provide subsumption constraints to distribute distributive features into coordinate f-structures.

Considering the limitations of the constraint solver, and in order to exploit more information for function annotation from a larger context than within the local tree, instead of generating f-structures via functional equations annotated to c-structure trees, I develop an alternative method which combines advantages of both the annotation-based and conversion-based approaches. The new method constructs an f-structure via an intermediate dependency structure that is directly converted from the original treebank or c-structure tree. The idea is based on the fact that the head nodes (annotated with the ↑=↓ equation) in a c-structure tree project to the same f-structure unit. The fact allows us to "fold" a c-structure tree into an intermediate, unlabelled dependency structure by collapsing the head nodes. The intermediate unlabelled dependency structure is more abstract and normalised

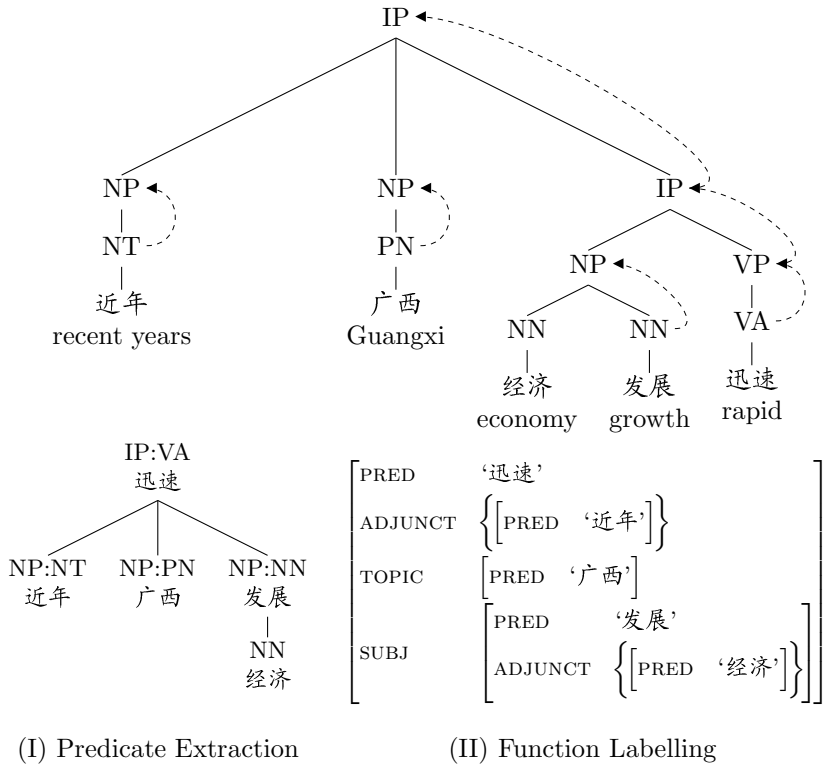


Figure 2.6: The conversion-based architecture for f-structure generation

compared to the original c-structure tree. Then the unlabelled dependency structure is used as input to a function assignment algorithm, which is similar to but simpler and more general than the conventional f-structure annotation algorithm directly operating on the original, more complex and varied c-structure trees.

The new conversion-based architecture is illustrated in Figure 2.6, which includes two major steps:

Predicate Extraction First, the algorithm identifies all predicates from the (local) c-structure tree, using head-finding rules similar to those used in the annotation-based approach to create a tripartition of each local subtree. By collapsing head-branches along the head-projection lines, the c-structure configuration is transformed into an intermediate unlabelled dependency structure, augmented with CFG category, POS and position information inherited from the c-structure.

Function Labelling Second, to create a labelled dependency structure, i.e. an LFG f-structure, the dependencies are assigned appropriate function relations using a reused and adapted revision of the left/right context annotation principles exploiting configurational, categorial, functional, and in addition, lexical information from each node of the intermediate unlabelled dependency tree.

By abstracting away from the "redundant" c-structure nodes in our intermediate dependency representations, the annotation principles can effectively apply to non-local sub-trees. This facilitates disambiguating different GFs in a larger context and resorting to lexical information. As a more abstract dependency structure is used to mediate between the c- and f-structure, the algorithm always generates an f-structure, and there are no clashing functional equations causing the constraint solver to fail. Moreover, the intermediate dependency structure can easily handle distribution into coordination structures by moving and duplicating the dependency branches associated with distributive functions. Furthermore, finite approximations of functional uncertainty equations resembling paths of non-local dependencies can also be computed on the intermediate dependency structure for the purpose of NLD recovery (see details in Chapter 3). Finally, in order to conform to the coherence condition and to produce a single covering and connected f-structure for every CTB tree, a post-processing step corresponding to the Catch-All and Clean-Up module in the annotation-based approach, is carried out to check duplicate functions and to catch and add missing annotations.

The conversion-based approach bears a broad similarity to the method developed in Frank (2000), with regard to the direct induction of f-structures from c-structures. However in Frank's algorithm, the mapping of c-structure to f-structure was carried out in one step using a tree/graph rewriting system. My method enforces a clear separation between constructing dependency structures and labelling function relations. The identification of predicates maps c-structure into an unlabelled dependency representation, and thus this component of my approach is designed particularly for a specific type of treebank encoding and data-structures. In contrast to Frank's approach, labelling function types is accomplished on the dependency representation

which is much more compact and normalised than the original c-structure representation, hence the function labelling rules are simpler and the overall architecture minimises dependency of the annotation rules on the particular treebank encoding. In a broader sense, this approach is more similar to the ones on automatical conversion between PS and DS representations (Lin, 1995; de Marneffe et al., 2006; Xue, 2007), for the purpose of state-of-the-art dependency parsing and evaluation.

2.5 Experiments and Evaluation

2.5.1 Development of a Gold Standard [6]

A set of gold-standard f-structures is necessary as a reference to evaluate the automatically acquired f-structures. The gold standard used in Burke et al. (2004) is based on 50 sentences randomly selected from articles 271-300 of an earlier version of the Penn Chinese Treebank — CTB2, which comprises exclusively Xinhua news. For the purpose of scaling up to the full CTB5.1, we split the CTB5.1 into 89 double-annotated files (see gold_standard_files distributed with CTB5.1) as test set,[7] 86 files as development set,[8] and the remaining 715 files as training set.[9] Each data set consists of all three (Xinhua, HKSAR and Sinorama) newswire sources. Then we randomly selected 200 sentences with length between 10 and 30 words from the CTB5.1 test set as a new gold standard and held out the old (Burke et al., 2004) 50 gold-standard sentences as a development set. The new gold standard is constructed semi-automatically: first, f-structures of the 200 sentences are automatically produced by the annotation algorithm, the resulting f-structures are then manually inspected and corrected in line with the Chinese LFG analyses.

[6] The work was carried out jointly with PARC (Palo Alto Research Center, Inc.).

[7] articles 1-40, 900-931, 550-554, 1018, 1020, 1036, 1044, 1060, 1061, 1072, 1118, 1119, 1132, 1141, 1142

[8] articles 271-325, 441-454, 545-549, 1019, 1073-1078, 1143-1147

[9] articles 41-270, 400-440, 600-885, 500-544, 590-596, 1001-1017, 1021-1035, 1037-1043, 1045-1059, 1062-1071, 1100-1117, 1120-1131, 1133-1140, 1148-1151

2.5.2 Experimental Results

To measure the quality of f-structures, f-structure units are broken down into a set of predicate-argument-adjunct (or dependency) relations, which are represented in the form of triples: $relation(predicate, argument/adjunct)$, as originally proposed by Crouch et al. (2002). And similar to Cahill et al. (2002) & Burke et al. (2004), I evaluate the automatically acquired f-structures quantitatively and qualitatively.

2.5.2.1 Quantitative Evaluation Results

Quantitative evaluation reflects the coverage of the acquired LFG resources by calculating the percentage of CTB trees producing a single connected and covering f-structure. In order to compare the improved annotation algorithm with the previous work, I first applied the f-structure annotation algorithm to CTB2, the same treebank as used in Burke et al. (2004). Table 2.1 provides a quantitative evaluation of the f-structures produced by the previous version[10] and the new version of the annotation-based algorithm. For the old annotation algorithm, 3 sentences did not produce any f-structure due to feature clashes in the function annotation, and 201 sentences output multiple f-structure fragments caused by nodes left unannotated by the old annotation principles. In contrast, the enhanced annotation algorithm generated connected and covering f-structures for all the 4,183 trees in CTB2.

	Burke (2006)		**Present Annotation**	
#f-structure	#sentence	percent	#sentence	percent
0	3	0.072	0	0
1	3979	95.123	4183	100
>=2	201	4.805	0	0

Table 2.1: Comparison of grammar coverage tested on CTB2

Table 2.2 gives a quantitative evaluation of the f-structures produced for all 18,804 trees in CTB5.1 by the annotation-based and conversion-based approaches presented in this dissertation. When applying the rule- and annotation-based algorithm to the larger-sized CTB5.1, feature clashes in the annotation of a small

[10]The latest results from Burke (2006) are given for the old annotation algorithm.

number of sentences (0.495%) result in no f-structure being produced for those sentences. By contrast, the conversion-based algorithm converts c-structures directly to the corresponding f-structures, obviating the constraint solver. As a result, it guarantees to produce a single, connected and covering f-structure for each treebank tree (although some f-structures are possibly not consistent as a consequence of erroneous function annotations). No unconnected, disjoint f-structure fragments are produced by either the annotation-based nor the conversion-based approach, because any nodes remaining unannotated are caught by default annotations in both methods.

	Annotation-Based		**Conversion-Based**	
#f-structure	#sentence	percent	#sentence	percent
0	93	0.495	0	0
1	18711	99.505	18804	100
>=2	0	0	0	0

Table 2.2: Comparison of grammar coverage tested on CTB5.1

2.5.2.2 Qualitative Evaluation Results

Qualitative evaluation compares the automatically generated f-structures against those in the gold-standard reference set in terms of precision, recall and the harmonic mean of precision and recall, viz. f-score:

$$Precision = \frac{number\ of\ correct\ triples\ in\ acquired\ f\text{-}structures}{number\ of\ all\ triples\ in\ acquired\ f\text{-}structures} \tag{2.1}$$

$$Recall = \frac{number\ of\ correct\ triples\ in\ acquired\ f\text{-}structures}{number\ of\ all\ triples\ in\ reference\ f\text{-}structures} \tag{2.2}$$

$$F\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.3}$$

Precision, recall and f-score are computed for two sets of triples, one (preds only) captures basic predicate-argument-adjunct relations, such as SUBJ, OBJ, TOPIC etc., and the other (all feats) also includes atomic-valued attributes/relations, such as NUM, PERS, CLAUSE-TYPE etc.

Table 2.3 compares the accuracy of the f-structures automatically acquired by my

annotation-based algorithm with those induced by Burke's algorithm, for the gold standard of 50 sentences.[11] The new annotation algorithm achieves an increase of nearly 10% in f-score for preds-only relations and 5% for all features. The substantial advances in the quality of the acquired f-structures are due largely to refinements of the annotation principles in my algorithm.

	Burke (2006)			**Present Thesis**		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Preds Only	81.44	86.29	83.79	94.09	92.14	93.10
All Feats	90.13	91.70	90.91	96.38	95.09	95.73

Table 2.3: Comparison of qualitative evaluation on development set

Table 2.4 reports the results evaluated against the new 200-sentence set of gold-standard f-structures for the annotation-based approach and the conversion-based approach. The results show no substantial difference in quality between f-structures acquired by annotating treebank trees or by directly converting treebank trees. This is reasonable as both versions of the algorithm are based on approximately the same annotation rules (head-finding rules and left/right context rules), which are implemented in different architectures. However, the quantitative evaluation results (Section 2.5.2.1) show that the conversion-based algorithm is more general and robust: it can be applied to generate dependency relations other than LFG f-structures and will always produce a single connected f-structure as a result of the transformation.

	Annotation-Based			**Conversion-Based**		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Preds Only	94.78	94.75	94.77	95.44	95.02	95.23
All Feats	96.23	96.47	96.35	96.34	96.46	96.40

Table 2.4: Comparison of qualitative evaluation on test set

[11] The results of the old annotation algorithm come from Burke (2006) evaluated against the old version of the 50-sentence gold standard and my algorithm against the new improved version.

2.6 Parsing into F-Structures

2.6.1 Parsing Architecture

So far, all experiments have been carried out exclusively on the existing CTB trees. In a real application, however, it is necessary to produce f-structures for new unseen text. Cahill et al. (2002) and Cahill (2004) designed and implemented two PCFG-based probabilistic parsing architectures for parsing unseen English text into f-structures: the *pipeline* and the *integrated* model. In the pipeline architecture, a PCFG is first extracted from the original treebank to parse new text into CFG trees or LFG c-structures. Then the most probable parser output trees are annotated with f-structure equations using the f-structure annotation algorithm. Finally the equations are collected and passed to a constraint solver to generate f-structures. In the integrated model, the original treebank trees are first annotated with f-structure equations and an annotated PCFG is extracted from the annotated version of treebank. The annotated PCFG can be viewed as an augmented version of a PCFG, where sequences consisting of CFG categories followed by one or more f-structure equations (e.g. NP[$\uparrow$SUBJ=$\downarrow$]) are treated as monadic categories for grammar extraction and parsing. Raw text is then parsed into f-structure-annotated trees by the annotated PCFG. And finally f-structure equations are collected to generate f-structures by the constraint solver.

Burke et al. (2004) applied both the pipeline and integrated parsing models to parse Chinese text into f-structures based on the CTB2. The integrated parsing model augments simple PCFG with the f-structure equations and effectively weakens PCFG independence assumptions. In this respect, the integrated model resembles a state-of-the-art treebank-based probabilistic parsing paradigm that improves parsing performance by, for example, grammar transformations (Johnson, 1998) or history-based models (Collins, 1999; Charniak, 2000; Klein and Manning, 2003; Petrov and Klein, 2007). Burke et al. (2004) reported that the integrated model resulted in a labelled f-score of 81.77% for parsing 300 sentences of length $\leq$40 in the

CTB2 test set evaluated using evalb,[12] contrasted with 78.78% labelled f-score for the pipeline model. Nevertheless, with regard to the quality of the final f-structures, the pipeline model achieved the best f-score of 83.89% for all f-structure features against the 50 gold-standard f-structures, outperforming the integrated model with an f-score of 82.12%. Furthermore, the pipeline parsing model is highly flexible because of the separation between c-structure parsing and f-structure annotation, which benefits from the two level representation (c-structure and f-structure) in the LFG formalism. In addition, the pipeline model provides the possibility to improve the PCFG parsing by exploiting external, state-of-the-art parsing technologies, and improving the f-structure annotation algorithm individually.

Considering the facts mentioned above, I choose the pipeline model to parse new text into f-structures based on CTB5.1. Unlike Burke et al. (2004) who used BitPar (Schmid, 2004) as the external PCFG parser, I adopt Bikel's parser (Bikel and Chiang, 2000) (a history-based, lexicalised and generative approach) in my parsing experiments, as it gives much better results than BitPar on Chinese data.

2.6.2 Experiments

The parsing model and f-structure annotation algorithm are applied to two sets of test data: (i) the gold standard set of 200 sentences selected from the CTB5.1 test set, and (ii) the full CTB5.1 test set of 1,913 sentences. For the gold standard, I compare the f-structures automatically produced by the parser for the 200 sentences with the manually corrected gold-standard f-structures. For the full test set, I evaluate the f-structures automatically acquired from the parser output trees against the f-structures acquired from the original treebank trees for the same sentences. As the conversion-based method is more robust and guarantees all sentences produce an f-structure, I apply this approach to convert the parser output (and treebank) trees to f-structures in these experiments. In all experiments, I use the CTB5.1 tokenisation and part-of-speech (POS) tags.[13] Table 2.5 provides the parsing results

[12]The tool is downloadable from http://nlp.cs.nyu.edu/evalb/

[13]Nevertheless, a POS tag that is supplied for a word is only used by the parser when that word was never observed in training; all other words are tagged according to pre-terminal rules learned

	200 Gold-Standard Sentences			All Test Sentences		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Labelled Bracket	83.30	80.46	81.86	78.42	75.17	76.76
Preds Only	69.01	59.07	63.66	67.54	55.94	61.19
All Feats	82.25	69.14	75.13	80.34	64.87	71.78

Table 2.5: Quality evaluation of f-structure acquired from parser output trees without CTB function tags

for both test sets.

It's a bit surprising that the qualitative f-structure results for the 200 gold sentences are higher than for all (1.9K) test sentences by about 2 percentage points for preds-only GFs and 3 percentage points for all features. The 200 gold-standard f-structures were hand-corrected, and therefore should be harder to get right for the parser than the purely automatically produced gold-standard f-structures for all test sentences. I conjecture that one of the most likely reasons is because the 200 sentences are restricted to between 10 and 30 words in length, which makes the data easier to parse. The first line of Table 2.5 gives the labelled bracketing scores of the output trees produced by Bikel's parser measured by evalb: an f-score of 81.86% is achieved for parsing the 200 gold-standard sentences, in contrast to 76.76% for all the test sentences. The 5% drop in the c-structure parsing contributes to a corresponding drop in f-structure evaluation.

Compared to the results given high-quality treebank trees as input (Table 2.4), it is not difficult to notice that the quality of the automatically produced f-structures drops sharply by about 32 percentage points in the preds-only f-score and 21 percentage points in the all-feats f-score given parser output trees as input. The drastic drop in the results on parser output trees is mainly due to labelled bracketing parser errors, but also because Bikel's parser (and most state-of-the-art treebank-based broad-coverage probabilistic parsers) produces neither CTB functional tags nor traces and co-indexation information, which are indeed present in the original CTB trees and greatly facilitate f-structure acquisition.

To partially solve the problem, following the approach described in Gabbard

during training.

et al. (2006), I made a small modification to Bikel's parser to allow it to retain the CTB functional tags in all the parameter classes. This extends Bikel's parser to include CTB functional tag labels as part of its output.

	200 Gold-Standard Sentences			**All Test Sentences**		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Labelled Bracket	81.93	79.52	80.71	77.10	74.25	75.65
Preds Only	75.57	66.54	70.77	72.21	61.58	66.47
All Feats	84.09	75.73	79.69	81.86	70.63	75.83

Table 2.6: Quality evaluation of f-structure acquired from parser output trees with CTB function tags

Table 2.6 gives results of the improved parsing model. Though the c-structures augmented with CTB functional tags produced by the modified Bikel's parser have slightly lower labelled bracketing f-scores compared to those produced by the original Bikel's parser, quality of the final f-structures converted from the function-labelled c-structures shows a substantial improvement, increasing f-scores by 4-7 percentage points on both test sets. This proves that the additional CTB functional tags play an important role in identifying appropriate functional relations.

Nevertheless, the results for parser output trees are still quite low compared to those for treebank trees, and there is a clear gap between the precision and recall scores for the resulting f-structures, which does not exist in the results for treebank trees (Table 2.4). One of the reasons for this is that non-local dependencies are not captured in the f-structures derived from the parser output trees (in other words the parser produce "proto" but not "proper" f-structures), whereas they are resolved by empty node and co-indexation information provided by the original treebank trees. This deficiency will be corrected by a postprocessing NLD recovery approach, presented in the next chapter.

2.7 Summary

This chapter has presented methods to automatically acquire wide-coverage, robust, probabilistic LFG resources for Chinese from the Penn Chinese Treebank.

Our starting point is the earlier, proof-of-concept work of Burke et al. (2004) on automatic f-structure annotation, and parsing for Chinese using the CTB2. We substantially extend and improve on this earlier research in regard to coverage, robustness, quality and fine-grainedness of the resulting LFG resources. We achieve this through (i) improving LFG analyses for a number of core Chinese phenomena; (ii) scaling the approach from 4.1K trees in CTB2 to 18.8K trees in CTB5.1; (iii) designing a new f-structure acquisition algorithm which integrates the general and linguistically-motivated left/right context annotation principles in the simpler and more robust conversion-based architecture. Against a new 200-sentence good standard of manually corrected f-structures, the method achieves 96.40% all-feats f-score for f-structures automatically generated from the original CTB trees and 79.69% all-feats f-score for f-structures generated from the trees output by Bikel's parser.

The results, while encouraging, can be improved given further concerted effort. For example, besides the categorial, configurational and function labels, fine-grained lexical information is necessary to identify subtle discrepancies in the same syntactic structures. A hard problem for Chinese (and also other languages such as English) is to correctly distinguish the OBL function from normal ADJUNCT for prepositional phrases, which in most cases is determined by the meaning of the main verb and the preposition. This would be an interesting issue worth exploring in the future but is not included in the present thesis. Another task that has been addressed, is to improve the quality of the f-structures derived from parser output trees. The next chapter will present one solution to this problem based on recovering non-local dependencies using finite approximations of functional uncertainty equations and LFG subcategorisation frames automatically acquired from the f-structure treebank converted from the CTB5.1 I have built in this chapter.

Chapter 3

Recovering Non-Local Dependencies

3.1 Introduction

The previous chapter presented treebank-based methods for the automatic acquisition of wide-coverage LFG resources for Chinese. The acquisition algorithm induces high quality f-structure resources given the manually annotated Penn Chinese Treebank (Table 2.4); however, it only produces basic and incomplete f-structures from parser output trees for raw text (Table 2.5). Although the drastic degradation in the quality of f-structures induced from raw text (compared to f-structures acquired from treebank trees) is primarily caused by syntactic parsing errors, it is also due to the fact that the output trees of most state-of-the-art probabilistic parsers are linguistically "impoverished": they do not include functional tags indicating (a limited number of) syntactic roles, nor do they capture Non-Local Dependencies (NLDs), via traces and coindexation as in the Penn Treebank. The f-structure acquisition algorithm, as described in Chapter 2, makes use of CTB functional tags and relies on traces and coindexation in the CTB to produce accurate and complete f-structures. In Section 2.6, I presented a simple method to preserve functional tags in the output parse trees by slightly modifying Bikel's parser. The retention of functional tags improves the quality of the automatically induced f-structures by 7 percentage

points in the preds-only f-score and 5 percentage points in the all-features f-score, evaluating against the 200-sentence good standard (Table 2.6). In this chapter, I describe a postprocessing approach to recover NLDs to further improve the basic "proto" f-structures generated from parser output trees to create fully NLD-resolved "proper" f-structures for raw text.

Section 3.2 provides a survey of NLD phenomena in Chinese as represented in the Penn Chinese Treebank. Section 3.3 reviews previous work on recovering NLDs on (i) the level of phrase structure (or c-structure) trees and (ii) dependency structures (or f-structures). Section 3.4 describes my substantial adaptation and extensions of the previous methodology of Cahill et al. (2004) originally developed for English, to tackle all types of NLDs in Chinese data. Section 3.5 presents experimental details and provides a dependency-based evaluation of NLD recovery. The training data for both Cahill et al. (2004) and my NLD-recovery algorithm is provided by treebank-based data. Section 3.6 presents an improved NLD-recovery algorithm where training instances are extracted from a set of "reparsed" data that is more similar to the imperfect f-structures acquired from parser output. An earlier version of some of the work presented in this chapter has been published as Guo et al. (2007b).

3.2 NLDs in Chinese

Non-local dependencies (also called unbounded, discontinuous or long-distance dependencies) occur in a substantial number of linguistic phenomena in Chinese, English and many other languages, such as topicalisation, relative clauses, raising & control constructions and coordinations. NLDs permit the location of the surface realisation of a constituent (referred to as "antecedent") to be different from the location where it should be interpreted semantically (referred to as "trace").

In this thesis I use the term "non-local dependencies" as a cover term for all missing or displaced elements represented in the CTB as empty categories (with or without coindexation), and the use of the term remains agnostic about fine-grained distinctions between non-local dependencies drawn in the theoretical linguistics lit-

(1) 不 愿意 发掘 培植 有 潜力 的 新 作家
not want look-for train have potential DE new writer
'(People) don't want to look for and train the new writers who have potential.'

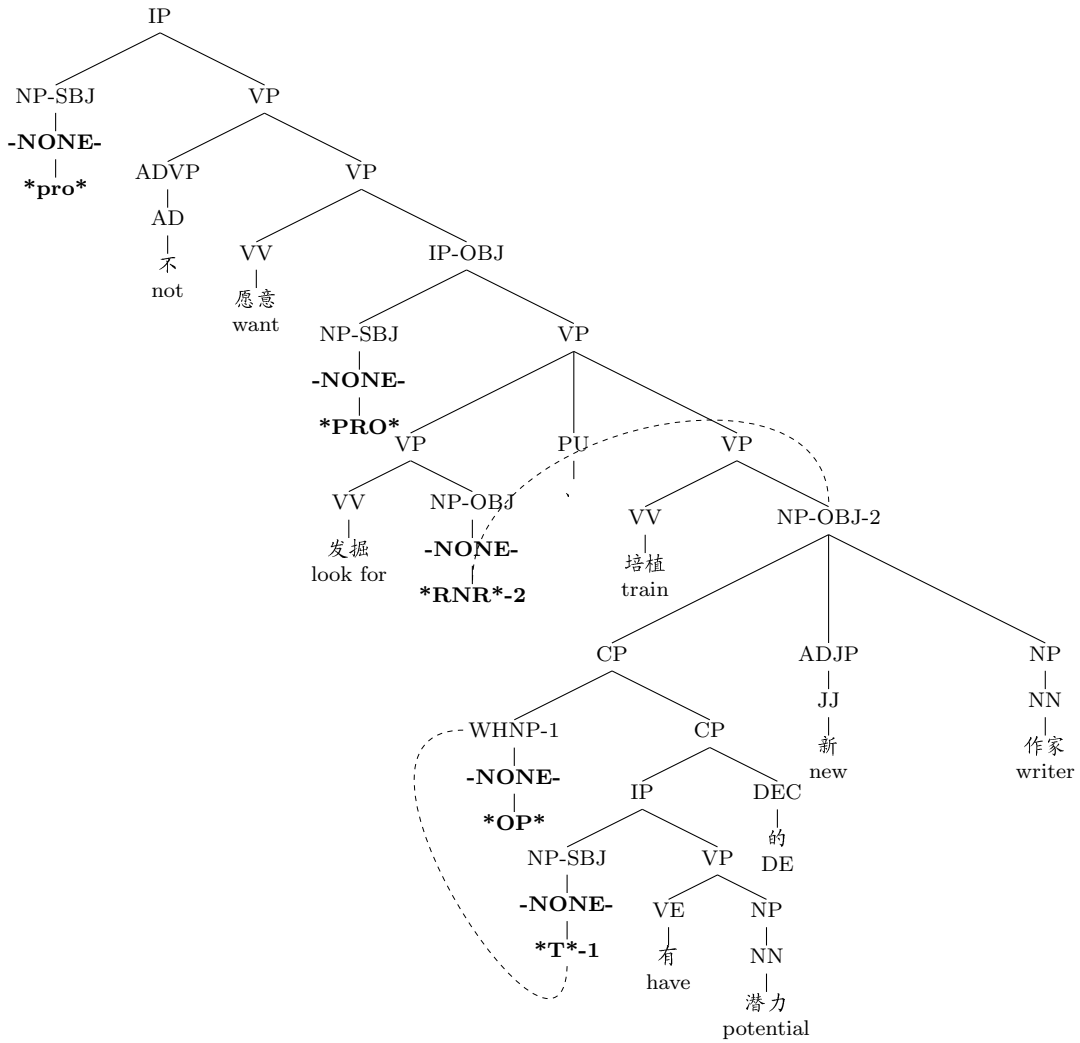


Figure 3.1: Example of NLDs represented in CTB, including dropped subject (**pro**), control subject (**PRO**), WH-trace in relativisation (**T**), and right node raising in coordination (**RNR**)

erature. NLDs are represented in the Penn Chinese Treebank in terms of empty categories (ECs) and (for some of them) coindexation with their corresponding antecedents, as exemplified in Figure 3.1. In order to give an overview of the characteristics of Chinese NLDs, I extracted all ECs and (where present) their coindexed antecedents from CTB5.1. Table 3.1 gives a breakdown of all types of ECs and their

	Label	Category	Antecedent	Count	Description (Example)
1	*T*	NP	WHNP	11670	WH traces (*OP*中国/China发射/launch**T***的/DE卫星/satellite)
2	*OP*	WHNP		11621	Zero relative pronouns (***OP***中国/China发射/launch的/DE卫星/satellite)
3	*PRO*	NP		10946	Control constructions (他/He设法/manage***PRO***脱身/get away)
4	*pro*	NP		7481	Pro-drop situations (***pro***不/not曾/ever遇到/encounter的/DE问题/problem)
5	*T*	IP	IP	575	Topicalisation (我们/we能/can赢/win，他/he说/say***T***)
6	*T*	PP	WHPP	337	WH traces (*OP*人口/population***T***密集/dense地区/area)
7	*OP*	WHPP		337	Zero relative pronouns (***OP***人口/population密集/dense地区/area)
8	*	NP	NP	291	Short-BEI constructions (我们/we被/BEI排除/exclude*在外/outside)
9	*RNR*	NP	NP	258	Coordinations (鼓励/encourage***RNR***和/and支持/support投资/investment)
10	*RNR*	CLP	CLP	182	Coordinations (五/five***RNR***至/to十/ten亿/hundred million元/Yuan)
11	*T*	NP	NP	93	Topicalisation (薪水/salary都/all用/use***T***来/for享乐/pleasure)

Table 3.1: Distribution of the most frequent types of ECs and their antecedents in CTB5.1

antecedents occurring more than 30 times. Together they cover 43,791 (99.6%) of the total 43,954 ECs in CTB5.1.[1]

According to their distinct linguistic properties, the empty categories in CTB5.1 are classified into three major types: zero relative pronouns, locally-mediated dependencies, and long-distance dependencies.

Zero Relative Pronouns (Table 3.1 lines 2, 7) are placeholders represented as (*-NONE- *OP**) to mark relative clauses. This EC type constitutes a special case as zero relative pronouns are local dependencies themselves, and thus are not coindexed with an antecedent. However, standing for the noun phrase/antecedent in the main clause that it modifies, a zero relative pronoun mediates non-local dependencies by linking the antecedent to one (argument or adjunct) GF of the verb embedded in the relative clause. Strictly speaking, there is no relative pronoun in Chinese and no relative clause is in fact introduced by a relative pronoun. Zero relative pronouns are only necessary for the appropriate semantic interpretation of relative clauses. For example, a zero relative pronoun is used to distinguish between a relative clause (2) where the object (**T**) of the verb inside the relative clause is "displaced" from the position where it is interpreted, and an appositive clause (3) which does not involve NLDs.

(2) *OP* 中国 发射 *T* 的 卫星
OP China launch *T* DE satellite
'The satellite that China launched'

(3) 他 离开 的 消息
he leave DE news
'The news that he left'

[1] An extensive description of the types of empty categories and the use of coindexation in CTB can be found in Section VI of the CTB bracketing guidelines.

Locally-Mediated Dependencies are also non-local dependencies, as they are projected through a third lexical item (such as a control or raising verb). Locally mediated dependencies hold between two adjacent levels and are therefore bounded. This type includes:

Short-BEI constructions (Table 3.1 line 8) which are roughly equivalent to passivisation in English. In a short-BEI construction, the original object of the verb in the active sentence is "fronted" to the subject position due to the presence of *被/BEI*.

Raising constructions as in (4) where the subject of the subordinate clause is realised syntactically as the subject of the matrix clause.[2]

(4) 他 好像 * 很 开心
he seems * very happy
'He seems very happy.'

Control constructions (Table 3.1 line 3) are indicated by the EC (*-NONE- *PRO**) and are coreferential with the subject or object of a "control" verb.[3] Besides subject and object control with definite reference, (*-NONE *PRO**) can also receive an arbitrary reading in some cases, where it can not be replaced by an overt antecedent. This is approximately analogous to unexpressed subjects of gerunds, imperatives and *to*-infinitive clauses in English, as in (5):

(5) 这里 不 许 *PRO* 抽烟
here not allow *PRO* smoke
'To smoke here is not allowed.'

Long-Distance Dependencies (LDDs) differ from locally-mediated dependencies in that there is no limit on how many layers of embeddings the relation between antecedent and trace spans. LDDs include the following phenomena:

[2]Raising constructions and short-BEI constructions are annotated identically as the EC (*-NONE- **) in CTB.

[3]Unfortunately, the CTB annotation does not always consistently coindex the locus/trace with its controller/antecedent for subject or object control constructions.

Wh-traces in relative clauses, where an argument (Table 3.1 line 1) or modifying adjunct (Table 3.1 line 6) of the verb in the relative clause "moves" and is coindexed with the "extraction" site.

Topicalisation (Table 3.1 lines 5 and 11) is one of the classic LDDs in English. In canonical LFG theory (Dalrymple, 2001), topicalisation is subject to the Extended Coherence Condition:

> TOPIC and FOCUS must be linked to the semantic predicate argument structure of the sentence in which they occur, either by functionally or by anaphorically binding an argument.

By contrast, as a topic-prominent language, Chinese does not require a fronted TOPIC
phrase to be constrained to an argument GF. For instance in (6), *北京/Beijing* is the TOPIC and the thing being talked about, but it is not related to any of the argument functions in the sentence.

(6) 北京 秋天 最 美
Beijing autumn most beautiful
'In Beijing autumn is the most beautiful season.'

Long-Bei constructions differ from short-BEI constructions in that the logical subject is explicitly expressed in the complement clause, and they allow long-distance movement as in (7).

(7) 张三 被 李四派 王五 *PRO* 打 了 *T*
ZhangSan BEI LiSi send WangWu *PRO* hit LE *T*
'LiSi sent WangWu to hit ZhangSan.'

Coordinations come into two types: (i) right node raising of an NP phrase which is an argument shared by the coordinated predicates (Table 3.1 line 9); and (ii) coordination of quantifier phrases (Table 3.1 line 10) or verbal phrases as in (8), where the verbal predicate inside one conjunct of the coordinated structure also bears the predicate function inside the other conjunct, although they each might take their own arguments or adjuncts.

(8) 我和 他 分别 去 公司 和 *RNR* 医院
I and he respectively go-to company and *RNR* hospital
'I and he went to the company and to the hospital.'

Pro-drop situations (Table 3.1 line 4) are widespread in Chinese because a subject or object is only semantically but not syntactically required to form a grammatical sentence. Dropped subjects or objects are indicated by the EC (*-NONE-* *pro**) in CTB. I treat pro-drop as a long-distance dependency as, in principle at least, the dropped constituent can be determined and replaced by an overt antecedent in the (often inter-sentential) context.

To compare the properties of Chinese NLDs as represented in the CTB with those in English, I also extracted all ECs and (where appropriate) coindexation from the Penn-II treebank for English data. Table 3.2 presents a quantitative comparison of NLD phenomena between Chinese and English. The numbers show that: (i) NLDs in Chinese are more frequent than in English (by nearly 1.5 times); and (ii) due to the high prevalence of pro-drop, 69% of the traces in Chinese are not explicitly linked to an antecedent, compared to 43% for English.

	#sentence	#EC	#EC/sent	#non-coindex	%non-coindex
Chinese	18,804	43,954	2.34	30,429	69.23
English	49,207	79,245	1.61	34,455	43.48

Table 3.2: Comparison of NLDs between Chinese data in CTB5.1 and English in Penn-II .

3.3 Previous Work

Non-local dependencies are common phenomena existing in many languages and crucial to the accurate and complete determination of semantic interpretation of language. Despite their importance, most treebank-based probabilistic parsers are trained and tested on a simplified or "shallow" version of tree representations without empty categories and coindexing information. As a result, they can only produce surface constituent trees recording purely local dependencies. It is by no means

a trivial task to recover NLDs and to translate shallow syntactic representations into proper predicate-argument structures or deep dependencies. During the recent years, there has been a growing interest in research on reconstructing non-local relationships either from output trees of state-of-the-art treebank-based parsers or using a more powerful syntactic framework, such as LFG, CCG, HPSG etc.

3.3.1 Recovering NLDs on Phrase Structure Trees

During the last decade, we have witnessed considerable efforts in capturing NLDs for Penn-style phrase structure trees by means of coindexing empty nodes with their antecedents. By and large, three types of approaches have been proposed, differing in how they interact with the wide-coverage probabilistic parser.

Post-Processing Methods reintroduce non-local dependencies in the impoverished parser output trees.

- Johnson (2002) presented the first post-processing method for inserting empty nodes and identifying their coindexed antecedents by a simple pattern-matching algorithm. The first phase of the algorithm extracts patterns — minimal connected tree fragments containing an empty node and all other nodes coindexed with it — from a training corpus of sections 2-21 of the Penn-II treebank. The second phase matches the extracted patterns against test trees without NLD representations (such as parser output trees). When a pattern matches, the algorithm introduces a corresponding non-local dependency by inserting an empty node and (possibly) coindexing it with a suitable antecedent.

- Higgins (2003), Levy and Manning (2004) and Gabbard et al. (2006) treat identification of NLDs as a classification task. Higgins (2003) focuses on one NLD type only, i.e. WH-traces, and describes a neural network classifier to search for a path from the surface location of the WH-constituent to the trace by a series of decisions, eventually locating the trace in the syntactic tree. Levy and Manning (2004) and Gabbard et al. (2006) describe a cascade of log-linear or maximum entropy classifiers, each of which utilises a wide range

of features selected for a certain subset of NLD types as represented in the Penn-II treebank. Levy and Manning's algorithm first identifies displaced nodes among the overt tree nodes and then determines an origin site within the tree (to insert an empty node) for each displaced constituent. By contrast, the algorithm of Gabbard et al. (2006) first inserts empty nodes in appropriate positions and then identifies the antecedents. Gabbard et al. (2006) report an f-score of 74.66% on recovering the most common types of NLDs for parsed sentences from section 23 of English Penn-II treebank produced by Bikel's parser. The result is competitive among all post-processing methods in the literature. Levy and Manning (2004), in addition, assessed the cross-linguistic effectiveness of their approach, comparing English with German, a less configurational language, in which NLD constructions (as expected) are more prominent and difficult to recover than in English.

- Unlike other corpus- and machine-learning-based approaches, Campbell (2004) makes use of the principles of Government and Binding theory underlying the annotation of the Penn Treebank. His algorithm deterministically detects empty nodes and finds their antecedents by sets of rules specifically designed for different types of empty categories. The rule-based method compares favourably to most machine-learning-based methods, exceeding them mainly by achieving higher recall on both empty category detection and antecedent identification, especially when evaluating on imperfect parser output trees.

In-Processing Methods integrate NLD recovery into the parser by enriching a simple PCFG model with GPSG-style *gap* features to mark the paths between traces and antecedents. This type includes:

- Model 3 of Collins (1999) which integrates the detection and resolution of WH-traces for relative clauses into a lexicalised PCFG. The grammar augments all nonterminals on the path linking the WH-extraction and the trace with an additional *gap* feature label.

- Schmid (2006) presents an unlexicalised PCFG parser that is automatically labelled with features in the style of Klein and Manning (2003), plus additional GPSG-like *gap* features. To generalise the *gap* features to all types of empty categories besides WH-traces, they are distinguished by the categories of the antecedents in different NLD types. The results reported in Schmid (2006) using standard Penn treebank training and test data, achieve an f-score of 84.1% for EC prediction and 77.4% for antecedent coindexation for trees produced by the unlexicalised PCFG parser, which exceed all previously reported results.

Pre-Processing Methods on the other hand, introduce empty nodes in the input string before parsing. This approach is presented in Dienes and Dubey (2003a,b). The algorithm first inserts empty nodes in the input sentence using a finite-state trace tagger, and produces a POS-tagged string with labelled empty nodes as input to a (lexicalised or unlexicalised) PCFG parser enriched with *gap* features. Antecedents are then identified on the parser output by threading the *gap* information. Dienes and Dubey (2003a) claim that the two-step approach outperforms the corresponding integrated method finding NLDs while parsing.

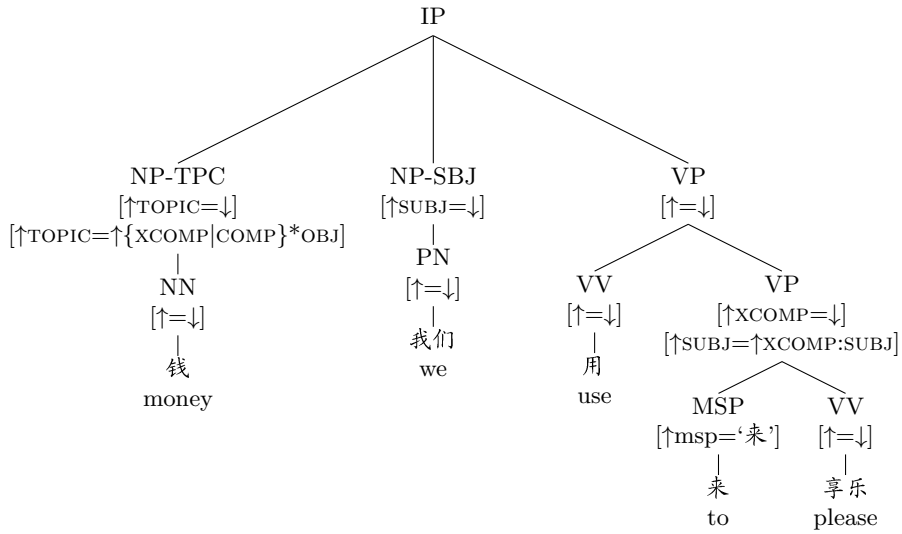
3.3.2 Recovering NLDs on Dependency Structures

Although phrase structure grammars are capable to implicitly represent NLDs with the aid of ECs and coindexation, NLDs are essentially semantic relationships. It is therefore arguably more natural and preferable to represent NLDs explicitly on a representation that captures the underlying predicate-argument structure. Such representations are provided by more expressive grammar formalisms, such as HPSG, CCG, TAG and LFG etc. With the recent progress in treebank-based automatic acquisition of these richer grammar resources, a number of statistical parsers based on TAG (Chiang, 2000), CCG (Hockenmaier, 2003), HPSG (Miyao et al., 2003) and Dependency Grammar (Nivre, 2006) incorporate non-local dependencies into their deep syntactic or semantic representations.

Besides integrated methods, post-processing approaches to NLD recovery have also been applied to dependency structures. Jijkoun (2003) describes a method similar to the pattern matching algorithm of Johnson (2002), but applied to dependency structures transformed from phrase structure trees. Due to the simplicity of dependency structures compared to constituent trees, the patterns are more general dependency graphs, and in consequence, the number of patterns extracted from the same training corpus are significantly reduced. The dependency-based pattern-matching algorithm achieves an improvement of approximately 10% in both precision and recall over the results of Johnson (2002). Jijkoun and de Rijke (2004) further improves the previous pattern-matching methods by performing a series of dependency graph transformations, including: changing dependency labels, adding new nodes and adding new dependencies. Each of the transformations is performed by a dedicated memory-based classifier trained on a dependency corpus derived from the Penn-II Treebank.

Couched in the formalism of LFG, Cahill et al. (2004) presents a post-processing method to recover non-local dependencies for the proto f-structures derived from treebank-based parser output. In LFG, NLDs are characterised at the level of f-structure by reentrancies via the mechanism of functional uncertainty (FU), obviating the need for traces and coindexation in c-structure trees. For the example in Figure 3.2, the reentrancy $\boxed{1}$ captures a topicalisation where the fronted topic *钱/money* is interpreted as the object of the predicate *用/use* in f_1, and $\boxed{2}$ indicates a subject control where the subject of the embedded predicate *享乐/please* in f_4 is controlled by the subject *我们/we* of the matrix clause. To specify the reentrancy $\boxed{1}$ in the f-structure, a functional uncertainty equation, i.e. a regular expression of the form [↑TOPIC=↑{XCOMP|COMP}*OBJ] is associated with the topic NP node, stating that the value of the TOPIC attribute is token identical with the value of an OBJ argument, reached through a path along any number (including zero) of the XCOMP or COMP attributes (accounting for a potentially unbounded length of the dependency). NLDs in LFG are also sensitive to subcategorisation frames (subcat frames) because of the coherence and completeness conditions that LFG imposes

(9) 钱 我们用 来 享乐
money we use to please
'Money, we use for pleasure.'



$$f_1\begin{bmatrix}\text{PRED} & \text{'用}\langle\text{SUBJ, OBJ, XCOMP}\rangle\text{'}\\ \text{TOPIC} & f_2\begin{bmatrix}\text{PRED} & \text{'钱'}\\ \text{NTYPE} & \text{common}\end{bmatrix}\boxed{1}\\ \text{SUBJ} & f_3\begin{bmatrix}\text{PRED} & \text{'pro'}\\ \text{PERS} & 1\\ \text{NUM} & \text{pl}\end{bmatrix}\boxed{2}\\ \text{OBJ} & \boxed{1}\\ \text{XCOMP} & f_4\begin{bmatrix}\text{PRED} & \text{'享乐}\langle\text{SUBJ}\rangle\text{'}\\ \text{SUBJ} & \boxed{2}\\ \text{MSP} & \text{'来'}\end{bmatrix}\end{bmatrix}$$

Figure 3.2: The formal mechanisms of functional uncertainty and reentrancy characterising NLDs in LFG

on f-structures: the within-clause argument grammatical function that a displaced constituent bears in an NLD must be subcategorised for by a predicate, and cannot already be instantiated by any other element in the local f-structure governed by that predicate. To model LFG NLDs, the algorithm of Cahill et al. (2004) automatically extracts subcat frames for each lemma and acquires finite approximations of FU equations linking reentracies from the f-structure treebank that is automatically acquired from the full English Penn-II trees with empty categories and coindexation. Given an unresolved NLD type (indicated by one of the antecedent discourse functions TOPIC, TOPIC-REL and FOCUS), NLD resolution asserts a reentrancy between the value of the NLD trigger (or antecedent) and a grammatical function (or trace) of an embedded local predicate, subject to the conditions that:

- the embedded grammatical function can be reached from the NLD trigger along an NLD path;
- the grammatical function is not already present at the level of embedding f-structure that terminates the NLD path;
- the local predicate subcategorises for the grammatical function in question.

All solutions satisfying the above conditions are then ranked according to the product of subcat frame and NLD path probabilities as in Eq. (3.1), where $P(s|w)$ is the probability of subcat frame s conditional on lemma w, and $P(p|a)$ is the probability of NLD path p conditional on triggering NLD antecedent a.

$$P(s|w) \times P(p|a) \tag{3.1}$$

3.4 NLD Recovery for CTB

The aim of the research presented in this chapter is similar to that of Cahill et al. (2004), viz. turning parser output proto-f-structures (as those produced in Section 2.6) into fully NLD-resolved proper f-structures. Therefore, I adopt the post-processing algorithm of Cahill et al. (2004) (henceforth C04 for short) as the basic

methodology, but substantially revisiting, adapting and extending it to all NLD phenomena and the specific requirements of the Chinese data as represented in the CTB (rather than a limited number of pre-defined types as in Cahill et al. (2004)).

3.4.1 The Core Algorithm

The C04 algorithm focuses on English and only resolves three types of NLDs with known antecedents indicated by certain grammatical functions, i.e. topicalisation (TOPIC), wh-movement in relative clauses (TOPIC-REL) and interrogatives (FOCUS). However, as illustrated in Section 3.2, except for relative clauses, antecedents in Chinese NLDs do not systematically correspond to types of GFs. Furthermore, more than half of all empty categories in the CTB are not explicitly coindexed with an antecedent due to the high prevalence of pro-drop in Chinese. Therefore, pre-defined antecedent GF types are not a reliable indicator to identify antecedents that trigger NLD resolution for Chinese. Instead, in my NLD-recovery algorithm I locate NLD traces using subcat frames for local predicates together with the LFG completeness and coherence conditions. Accordingly, the C04 algorithm is modified and substantially extended as follows:

1. I extract NLD resolution paths p linking reentrances (including empty paths without an antecedent) from the proper f-structures automatically generated from the full CTB trees of the training set. To better account for all Chinese NLD types present in the CTB, I learn the probability of p conditioned on the GF associated with the trace t (instead of the antecedent a as in the C04 algorithm). The path probability $P(p|t)$ is estimated as Eq. (3.2), and some probabilistic path examples extracted from CTB5.1 are listed in Table 3.3.

$$P(p|t) = \frac{Count(p,t)}{\sum_{i=1}^{n} Count(p_i,t)} \tag{3.2}$$

2. From the training set of the proper f-structure bank derived from CTB5.1, I extract subcat frames s for each verbal predicate w. In contrast even to English, Chinese has very little morphological information, and, as a con-

Trace	Path	Probability
ADJUNCT	(↑TOPIC-REL)	0.9018
ADJUNCT	(↑COORD TOPIC-REL)	0.0192
ADJUNCT	NULL	0.0128
...	...	...
OBJ	(↑TOPIC-REL)	0.7915
OBJ	(↑COORD COORD OBJ)	0.1108
...	...	...
SUBJ	NULL	0.3903
SUBJ	(↑TOPIC-REL)	0.2092
...	...	...

Table 3.3: Examples of probabilistic NLD resolution paths

sequence, every word in Chinese has a unique form regardless of its varied syntactic (part-of-speech) distribution. To disambiguate subcat frames, I use two more syntactic features w_feats in addition to word form so as to correctly choose the most appropriate frame in a particular context. For a given word w, w_feats include:

- w_pos: the part-of-speech of w
- w_gf: the grammatical function of w

The probability of a lexical subcat frame s is then calculated conditional on the word w in combination with its features w_feats:

$$P(s|w, w\_feats) = \frac{Count(s, w, w\_feats)}{\sum_{i=1}^{m} Count(s_i, w, w\_feats)} \tag{3.3}$$

As more conditioning features may cause data sparseness, I smooth the subcat frame counts $Count(s, w, w\_feats)$ by a backoff ignoring the particular word form according to Eq. (3.4), so as to increase the coverage of the automatically acquired subcat frames. $P(s|feats)$ is the probability of the subcat frame for each feature set {pos, gf}, which is estimated as Eq. (3.5), and is weighted by

a constant parameter λ in the smoothing.

$$Count_{sm}(s, w, w\_feats) = Count(s, w, w\_feats) + \lambda P(s|feats) \quad (3.4)$$

$$P(s|feats) = \frac{Count(s, feats)}{\sum_{i=1}^{k} Count(s_i, feats)} \quad (3.5)$$

The final subcat frame probabilities are estimated from the smoothed frequencies as in Eq. (3.6). Some automatically extracted subcat frames and their probabilities for the verb *有/have* are listed in Table 3.4.

$$P_{sm}(s|w, w\_feats) = \frac{Count_{sm}(s, w, w\_feats)}{\sum_{i=1}^{m} Count_{sm}(s_i, w, w\_feats)} \quad (3.6)$$

Word|POS|GF	**Subcat Frames**	**Probability**
有|VE|ADJ-REL	[subj, obj]	0.6769
有|VE|ADJ-REL	[subj, comp]	0.1531
有|VE|ADJ-REL	[subj]	0.0556
...	...	...
有|VE|COMP	[subj, obj]	0.4805
有|VE|COMP	[subj, comp]	0.2587
...	...	...
有|VE|TOP	[subj, comp]	0.4397
有|VE|TOP	[subj, obj]	0.3510
...	...	...

Table 3.4: Examples of automatically extracted probabilistic subcat frames

3. Given the set of subcat frames s for the word w, and the set of paths p for the trace t, the NLD recovery algorithm traverses the f-structure f to:

   - predict a displaced argument t at a sub-f-structure h by comparing the local PRED w to w's subcat frames s: t can be inserted at h if h together with t is complete and coherent with regard to subcat frame s
   - traverse f inside-out starting from t along path p
   - link t to antecedent a if p's ending GF a exists in a sub-f-structure in f reached from t along path p; or leave t without an antecedent if an empty path for t exists.

4. Finally, resolution candidates are ranked according to the product of subcat frame and NLD path probabilities:

$$P_{sm}(s|w, w\_feats) \times \prod_{j=1}^{k} P(p|t_j) \tag{3.7}$$

As, apart from the maximum number of arguments in a subcat frame, there is no a priori limit on the number of missing GFs or traces in a local f-structure, the second product in Eq. (3.7) is itself a product over local traces. For example the verb *发掘/look for* in Figure 3.1 subcategories for a displaced SUBJ controlled by the matrix subject and a displaced OBJ that moves to the object position of the right coordinated verb *培植/train*. The algorithm ranks resolutions with the product of the path probabilities of each (of k) missing argument GF(s).

3.4.2 A Hybrid Strategy

As described in Section 3.2, there are three major types of NLDs in the CTB that present different linguistic properties. This requires a more fine-grained strategy than the general one described so far. Moreover, as the NLD recovery method described above is triggered by "missing" subcategorisable GFs, cases of NLDs in which the trace is not an argument of the local f-structure, but an ADJUNCT or a TOPIC in a relative clause or an empty PRED in a verbal coordination, can not be recovered by the basic algorithm. Table 3.5 gives a summary of the types of Chinese NLDs that can be recovered by C04 and the algorithm presented in Section 3.4.1, represented in terms of GF types associated with the NLD traces and antecedents.[4]

Table 3.5 shows that both algorithms have limitations on certain NLD types, although the modified algorithm is capable of recovering a wider variety of NLDs. A further intuition is that it should be easier to relocate the origin of an already established NLD antecedent (as in C04) than to determine both the antecedent

[4] *None* indicates the NLD type without an overt antecedent, such as pro-drop situations; PRED indicates the trace itself is a predicate governing its own arguments or adjuncts, for example coordination constructions where two conjuncts share the identical predicate, as in sentence (8) in Section 3.2.

	Antecedent			Trace		
	TOPIC-REL	Others	None	Argument	Adjunct	PRED
Cahill et al. (2004)	√			√	√	
Present Thesis	√	√	√	√		

Table 3.5: Comparison of the capability for recovering Chinese NLDs between C04 and the modified algorithm

and trace in some as yet unknown type of NLD (as in the approach developed in Section 3.4.1). For these reasons, I design and implement a hybrid method to resolve all types of NLD in the CTB and to maximise the accuracy of the algorithm. The fine-grained approach involves four steps to deal with different NLD types:

1. Applying a few simple heuristic rules to insert:

   - empty PREDs for coordination constructions in which conjuncts share the predicate;
   - zero relative pronouns for relative clauses, which is triggered by the GF ADJ-REL in our system.

2. Inserting an empty node with GF SUBJ for the short-BEI construction, control and raising construction (indicated by the GF XCOMP in our system), and relate it to the OBJ (if there is one) or SUBJ in the upper-level f-structure.

3. Exploiting the C04 algorithm, which conditions the probability of NLD paths on a given antecedent rather than on some hypothesised trace, to resolve WH-traces in relativisation, including those bearing the ungovernable GFs TOPIC and ADJUNCT.

4. Using the modified algorithm (presented in Section 3.4.1) to resolve the remaining and the most typical NLD types for Chinese.

This hybrid approach now covers all cases identified in Table 3.5.

3.5 Experiments and Evaluation

3.5.1 Experimental Setup

Experiments are carried out on the CTB5.1 that is split into the same training, development and test sets as in the previous chapter. In order to learn probabilistic NLD paths and subcat frames, the full-fledged treebank trees of the training set are converted into NLD-resolved f-structures using the conversion-based algorithm described in Section 2.4. From the resulting Chinese f-structure bank, a total of 19,133 different types of lexical subcat frames and 245 types of NLD paths are extracted. I test the NLD recovery models against two different kinds of input data: first, perfect input, i.e. the proto-f-structures derived from CTB gold-standard trees stripped of all empty nodes and coindexation; and second, imperfect input, viz. the proto-f-structures induced from output trees of Bikel's parser.

The most widely-used metric for evaluating NLD recovery was originally proposed by Johnson (2002). The metric judges an EC inference correct if it matches the gold standard in its syntactic category and string position (indicated by the left and right boundaries). However, as some researchers (Campbell, 2004; Levy and Manning, 2004) have noted, this metric leads to both false positives and false negatives when inserting an EC at a wrong position in the parse tree (but a correct position in the string). And more importantly, the purpose of inserting empty categories into trees is to recover semantic information such as predicate-argument relations, which is, however, not measured by the string-based evaluation metric. I therefore adopt the stricter dependency-based evaluation method as used to measure the quality of automatically acquired f-structures in Chapter 2. In this evaluation, a trace is represented as a triple of the form $relation(\textsc{pred} : location, \textsc{trace} : location)$, where *relation* denotes the predicate-argument-adjunct relationship between the inserted trace and the local predicate, and *location* is the string position of each word in the input sentence.[5] Following most previous work, the evaluation consists of two measures: first, to evaluate whether a trace is inserted at an appropriate

[5] An EC does not take an extra position, it has the same string index as the immediately preceding non-empty word.

location (insertion), in a triple for this measure, the TRACE is represented by the generic empty category, for the example from Figure 3.2, OBJ(用/use:3, -NONE-:3) indicates that a trace bearing the OBJ function is inserted immediately after the predicate *用/use*; second, to evaluate whether the trace is correctly related to its corresponding antecedent (recovery), in this measure the TRACE is instantiated by the related antecedent, for example, the reentrancy 1 in Figure 3.2 is represented as OBJ(用/use:3, 钱/money:1). Again, precision, recall and f-score are calculated for the evaluation.

3.5.2 Experimental Results

Tables 3.6 and 3.7 summarise the results for trace insertion and antecedent recovery testing on stripped CTB trees and parser output trees, respectively. To facilitate comparison with the approach of Cahill et al. (2004), in addition to the basic and hybrid algorithms described in the previous sections, I also implemented the C04 algorithm on the Chinese data and evaluated the results.

CTB	**Insertion**			**Recovery**		
Trees	Precision	Recall	F-Score	Precision	Recall	F-Score
C04	**96.49**	52.18	67.73	**92.18**	49.85	64.71
Basic	95.71	89.06	92.27	70.04	65.17	67.52
Hybrid	95.88	**89.88**	**92.79**	83.86	**78.61**	**81.15**

Table 3.6: Evaluation of trace insertion and antecedent recovery on stripped CTB trees

Parser	**Insertion**			**Recovery**		
Output	Precision	Recall	F-Score	Precision	Recall	F-Score
C04	**74.22**	38.65	50.83	**62.21**	32.40	42.61
Basic	74.02	62.69	67.89	48.28	40.89	44.28
Hybrid	74.03	**63.02**	**68.09**	57.40	**48.87**	**52.79**

Table 3.7: Evaluation of trace insertion and antecedent recovery on parser output trees

The results clearly show that the C04 algorithm has the lowest recall due to its restriction to certain types of NLD phenomena. On the other hand, as expected, the C04 algorithm achieves the highest precision, especially for antecedent recov-

ery. This is because the C04 algorithm identifies the original location (trace) for a known antecedent, whereas my modified algorithm tries to identify both the trace and antecedent (including null antecedent) for an unknown NLD type. In order to recover more NLD types with maximum accuracy, the hybrid model combines the C04 algorithm with the basic algorithm, viz. it recovers certain NLD types with known GF(s) (TOPIC-REL in the CTB data) using the C04 algorithm triggered by antecedents, and recovers unknown NLDs using the basic algorithm triggered by hypothesising traces. However, the two algorithms may conflict during NLD resolution in certain situations: (i) inserting a trace at the same site but relating it to different antecedents, or (ii) relating the same antecedent to different traces. Because the C04 algorithm tends to be more precise for known antecedents than my basic algorithm, the hybrid method keeps the traces inserted by the C04 algorithm and abandons those inserted by the basic algorithm in the case of conflict. The hybrid method evidently outperforms the C04 algorithm for both trace insertion and antecedent recovery due to its wider coverage of NLD types. It also substantially improves the performance of the basic model for the task of antecedent recovery by taking advantage of the C04 algorithm.

Antecedent Recovery	**Basic Model**			**Hybrid Model**		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Overall	70.04	65.17	67.52	**83.86**	**78.61**	**81.15**
SUBJ	61.39	58.87	60.10	80.64	77.47	79.02
OBJ	64.48	58.50	61.34	76.37	49.23	59.87
ADJUNCT	0.00	0.00	0.00	16.67	3.53	5.83
TOPIC	0.00	0.00	0.00	57.38	66.67	61.67
COORD	0.00	0.00	0.00	93.75	88.24	90.91
TOPIC-REL	98.87	98.27	98.57	98.87	98.27	98.57

Table 3.8: Breakdown by major grammatical functions for antecedent recovery on stripped CTB trees

Table 3.8 gives a breakdown by major GFs associated with the NLD traces of the results for antecedent recovery by the basic and hybrid models, testing on stripped CTB trees. Compared to the generic strategy that indiscriminately resolves different NLD types, the fine-grained hybrid method is sensitive to particular linguistic

properties of different NLD types. As a result, it:

- is capable of recovering empty predicates in verbal or quantifier coordination constructions (COORD);
- can relate the TOPIC-REL antecedent in a relative clause to a trace bearing an ungovernable GF ADJUNCT or TOPIC;
- substantially enhances the recovery of SUBJ-type traces by separating WH-traces and locally-mediated dependencies from other LDDs.
- also increases the precision of OBJ-type traces at a cost of lower recall. Examining the development data, I found that the lower recall is to some extent relevant to the correct recovery of SUBJ-type traces. Since the subject in Chinese has a very strong tendency to be dropped if it can be inferred from context, the empty NLD path has the greatest probability in all resolution paths conditional on SUBJ, and this prevents the SUBJ-type trace from finding an appropriate antecedent in the basic model. This results in the fact that the remaining WH-traces are identified as bearing the OBJ function, which contributes to the higher recall of the OBJ-type traces. However, the number of these cases is considerably reduced in the hybrid model, which correctly resolves some of the WH-traces as SUBJs by the C04 algorithm based on the NLD paths conditional on known types of antecedents rather than traces.

In conclusion, the hybrid methodology capturing specific linguistic properties of various NLD types, greatly improves the performance for the antecedent recovery task over the generic method, increasing the f-score by 13.6 percentage points on CTB trees and 8.5 percentage points on parser output trees.

3.6 Better Training for Parser Output

3.6.1 Motivation

The experiments reported in Section 3.5 show that although the NLD recovery algorithm achieves encouraging results for perfect input CTB trees, it is sensitive to the

noise in trees produced by Bikel's parser, incurring a dramatic performance drop of 24.6 percentage points for trace insertion and 28.3 percentage points for antecedent recovery. The same trend was also reported in the literature on NLD recovery for English data: Campbell (2004), for example, reported a drop from 93.7% to 76.7% for insertion and from 89.0% to 70.8% for recovery when moving from the Penn-II treebank trees to the output of Charniak's parser (Charniak, 2000). The lower results obviously reflect errors introduced into the output trees during parsing. My NLD recovery algorithm runs on f-structures automatically generated by the acquisition algorithm (as presented in Chapter 2), which is highly tailored to the CTB coding scheme (using configurational, categorial and functional tag information) and suffers considerably from errors produced by the parser. As indicated by the results in Tables 2.4 and 2.6, the quality of the automatically acquired f-structures decreases sharply from an f-score of 96.40% to 75.83%, evaluating all features given parser output trees (even augmented with functional tags). There is no doubt that the poor quality of the parser-based proto-f-structures contributes to the performance degradation for NLD recovery on such f-structures.

Ideally, methods based on machine learning techniques should train models on data closely resembling the final test instances. However, in the NLD recovery task, NLD resolution models are trained on proper f-structures generated from the original CTB trees, whereas at run time the models operate on proto-f-structures automatically generated from noisy trees output by Bikel's parser. This constitutes a serious drawback as regards the machine-learning approach: the training instances derived from perfect treebank trees are substantially different from the test data derived from imperfect parser output trees that contain a certain proportion of errors. This observation motives a method to reduce the difference between training and test instances by using parser output trees rather than treebank trees to train the NLD recovery models.

3.6.2 Methods

In order to make the training material more similar to the parser-based test data on which the NLD recovery model operates, I parsed the training data by Bikel's parser, and extracted NLD paths and subcat frames from the "reparsed" data in four steps:

Reparsing the training portion of the treebank. To avoid running the parser on its training data, I carried out 10-fold cross-validation, dividing the training data into 10 parts and parsing each part in turn with the parser trained on the remaining 9 parts.

Converting the parser output trees into imperfect and incomplete proto-f-structures, which possibly contain parse errors and do not capture NLDs via reentrancies, as empty nodes and coindexation are ignored by the parsing model.

Restoring reentrances in the imperfect f-structures. To do this, I convert the original CTB trees of the training set into proper f-structures, and map the NLDs from these gold f-structures to the corresponding f-structures induced from parser output trees. Since the parser-output f-structures contain errors, they are usually not identical to the gold f-structures. I match a local parser-output f-structure with a local gold f-structure on condition that: (i) the values of their PRED attributes are identical; (ii) their string positions in the input sentence are equivalent. The NLD restoration algorithm traverses the gold f-structure:

1. finding a local f-structure f where a reentrancy representing the trace t exists;
2. mapping f to the corresponding f' of the parser-output f-structure (if there is), and inserting a trace t' in f' if t' preserves the uniqueness condition of f';
3. identifying the antecedent reentrant f-structure a in the gold f-structure, and mapping a to the corresponding local f-structure a' in the parser-output f-structure (if there is);

4. linking the local f-structures a' and t' in the parser-output f-structures (i.e. establishing the reentrancy between a' and t').

Extracting NLD resolution paths linking reentrancies and subcat frames of all local predicates from the parser-output f-structures.

3.6.3 Results

The restoration algorithm described above recovered 35,581 (approximately 81%) reentrancies for the f-structures derived from parser output trees of the CTB5.1 training set. However, not all reentrancies can be retrieved, about 6% of them are missing because of parsing errors which lead to mismatches between the gold and the parser-output f-structures, and the remaining 12% unrecovered reentrancies are due to violations of the uniqueness condition when inserting reentrancies. Even so, a total of 1,068 distinct types of NLD paths are extracted from the f-structures derived from the *reparsed* data of the training set, which is about 4.4 times (of 245 types) that obtained from the gold f-structures. This fact implies that the f-structures derived from imperfect parser output trees manifest a greater diversity than those derived from the original treebank trees. To verify the prediction that the instances extracted using the *reparsing* method are more similar to test instances, I compare the overlap of NLD paths extracted from parser-output and gold f-structures between the development set and the training set. As shown in Table 3.9, although the standard training set covers 84.4% NLD path types of the development set derived from gold treebank trees, it has a rather low coverage (16.3%) on the development data derived from parser output trees. By contrast, the *reparsing* training method boosts the coverage to 62.3% even from the portion of reentrancies that are recovered by the restoration algorithm.

Results of NLD Recovery

I repeat the experiments recovering NLDs for the parser-based proto-f-structures of the test data as described in Section 3.5. Table 3.10 compares the results for trace

NLD paths		Gold	Parser-Output
Development Set	–	77	252
Standard Training	245	65(84.4%)	41(16.3%)
Reparsing Training	1,068	50(64.9%)	157(62.3%)

Table 3.9: Count and overlap of NLD paths against the development set for the two training methods

insertion and antecedent recovery by the hybrid approach using: (i) the *standard* method to train models on perfect f-structures derived from gold CTB trees (repetition of Table 3.7); and (ii) the *reparsing* method to train models on imperfect f-structures derived from parser output trees. The results show that the *reparsing* training method does outperform the *standard* training but the difference is fairly modest. To measure the significance of improvements in the scores, I perform a paired t-test on the mean difference of the f-scores. For the trace insertion, I obtain a p-value of 0.006, and for the antecedent recovery the p-value is 0.03, showing that the improvements in both tasks are statistically significant at the 95% level.

	Insertion			**Recovery**		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Standard	74.03	63.02	68.09	57.40	48.87	52.79
Reparsing	74.32	63.33	68.38	57.54	49.03	52.94

Table 3.10: Evaluation of trace insertion and antecedent recovery by hybrid models trained on gold-standard f-structures and parser-output f-structures

One reason why the reparsing approach does not demonstrate a more pronounced increase in scores over the standard method is apparently because of parsing errors. The core NLD recovery algorithm presented in the thesis is driven by "missing" arguments, however as indicated by the training data, there are more than 12% local predicates that are saturated by mistake, and there is no means to recover NLDs for the f-structures governed by those predicates. Another likely reason, I speculate, is related to the relatively small amount of training material used. Compared to 245 NLD path types extracted from nearly 44,000 reentrancies, the reparsing model with 1,068 NLD path types learned from 35,581 reentrancies may suffer from data sparseness. And moreover, the great diversity presented in the f-structures induced

from parser output trees results in relatively low path coverage where nearly 40% NLD paths in the parser-output development set can not be covered by the 1,068 NLD path types, contrasted with the fact that 245 NLD path types from the standard training set cover more than 80% NLD paths of the development data coming from the original CTB trees.

Results of F-Structure Parsing

The ultimate goal of the research presented in this chapter is to translate the parser-output proto-f-structures into NLD-resolved proper f-structures. I now investigate to which degree the quality of the automatically induced f-structures can be improved by reintroducing reentrancies to capture NLDs. As in Section 2.6, I carry out a qualitative evaluation of NLD-resolved f-structures, which are automatically acquired for raw text by the pipeline parsing model in conjunction with the post-processing NLD recovery component. Again, two sets of test data are used: (i) the manually corrected f-structures of the 200-sentence gold-standard test set; and (ii) the 1,913 f-structures acquired from original CTB trees (with empty nodes and coindexation) of the CTB5.1 test set. Table 3.11 summarises the evaluation results for the f-structures before and after recovering NLDs. There is a clear increase in the recall scores due to the recovery of reentrancies representing NLDs, while the precision scores decrease slightly. Overall, the NLD recovery component gives an improvement in the f-score by about 3 percentage points evaluating preds-only GFs and about 2 percentage points evaluating all features against both test sets.

	200 Gold-Standard Sentences			**All Test Sentences**		
-NLD	Precision	Recall	F-Score	Precision	Recall	F-Score
Preds Only	75.57	66.54	70.77	72.21	61.58	66.47
All Feats	84.09	75.73	79.69	81.86	70.63	75.83
+NLD	Precision	Recall	F-Score	Precision	Recall	F-Score
Preds Only	75.25	73.07	74.15	71.13	67.40	69.22
All Feats	83.53	80.67	82.08	80.80	75.27	77.94

Table 3.11: Comparison of f-structures before and after recovering NLDs

Up to now, Chinese parsers based on deep and wide-coverage grammars are very

rare. One and possibly the most suitable system to compare against is the PARC XLE multi-language parser (Crouch et al., 2006) that uses hand-crafted LFG grammars. Fang and King (2007) report on progress of the Chinese grammar developed for the XLE parsing system. Parsing the jointly developed 200 gold-standard test sentences with PARC's Chinese grammar (as of March, 2007), 188 sentences receive full parses, resulting in an f-score of 72.7% evaluating preds-only GFs (Table 3.12). The treebank-based automatically acquired resources achieve 100% coverage and a preds-only score of 74.2%. However it is important to note that the experiments carried out in Fang and King (2007) include automatic segmentation and POS tagging using a tokeniser developed by Beijing University, which is based on a scheme somewhat different from the CTB gold standard. By contrast, I adopt the CTB segmentation and provide POS tags for unknown words in the parsing experiments. This means that the results cannot be directly compared. However, it is evident that the treebank-based automatically acquired Chinese LFG resources are strongly competitive with respect to coverage, quality, and development time.

	Coverage	Precision	Recall	F-Score
PARC XLE	94%	73.1	72.4	72.7
Present Thesis	100%	75.3	73.1	74.2

Table 3.12: Comparison of f-structures acquired by hand-crafted and treebank-induced grammars

3.7 Summary

In this chapter, I investigated the problem left over from the previous chapter for parsing new text into f-structures: as most state-of-the-art statistical parsers ignore non-local dependency relationships recorded by means of empty categories and coindexation in the Penn Treebanks, the LFG resources automatically acquired for raw text by the pipeline parsing model are only basic, incomplete proto-f-structures with NLDs unresolved. For turning the proto-f-structures into NLD-resolved proper f-structures, I presented a post-processing approach to reintroduce reentrancies into f-structures to capture NLDs that were originally overlooked by the parsing models.

Based on a thorough inspection of all NLD types in the CTB, my NLD recovery algorithm involves hybrid strategies including:

- a number of heuristic rules to tackle the locally-mediated dependencies and particular types of NLDs, such as zero relative pronouns and null predicates in coordinations;
- two statistical models to tackle long-distance dependencies based on probabilistic NLD paths linking reentrancies and subcategorisation frames for local predicates, which are automatically acquired from the f-structure resources induced from the original CTB trees.
    - One statistical model (Cahill et al., 2004) learns probabilities of NLD paths conditional on GFs associated with known antecedents to more precisely identify traces for the known NLD type — WH-movement in relative clauses;
    - The other statistical model, by contrast, learns probabilities of NLD paths conditional on GFs associated with traces to recover the remaining NLDs, including the types in which the trace is not related to an overt antecedent, such as pro-drop in Chinese.

Another contribution of this chapter consists in presenting a theoretically sound method of training on imperfect parser-output f-structures rather than gold f-structures obtained from the original treebank trees for the NLD recovery task. The reparsing training method enhances the similarity of the training materials to the test instances, and modestly yet still significantly improves the final performance of the NLD recovery models. A similar scenario is described in Chrupała et al. (2007), which present related ideas for improving functional labelling of parser output trees. They show that extracting training instances from the reparsed training part of the treebank results in better training material and achieves statistically significantly higher f-scores on the function labelling task for the English Penn Treebank. However, the function labelling results reported in Chrupała et al. (2007) for the Penn

Chinese Treebank do not show any statistically significant difference between the normal training and the better training methods. To the best of my knowledge, so far nobody has attempted to apply this better training method to the NLD recovery task.

The final evaluation results demonstrate that the NLD recovery algorithm effectively turns the shallow f-structures acquired in Chapter 2 into linguistically rich and deep LFG representations. After recovering NLDs, the quality of the f-structures shows a considerable improvement, achieving an f-score of 74.15% for preds-only GFs and 82.08% for all features evaluating against the gold-standard test set of 200 f-structures for Chinese.

Part II

LFG-Based Generation

Chapter 4

Introduction to Natural Language Generation

This dissertation has provided a method for treebank-based automatic acquisition of wide-coverage Chinese LFG resources (in Chapter 2) and parsing Chinese into f-structures, including recovering non-local dependencies (in Chapter 3) with such resources. As a complementary operation to parsing, natural language generation is an important component in natural language applications such as machine translation, text summarisation, question answering and dialogue system, among others. While parsing is possibly one of the most extensively studied areas of natural language processing, generation is relatively under-developed and there remains ample room to make improvements. In the following three chapters, I will explore the task of Chinese generation in the framework of LFG, specifically, realising sentence surface forms from LFG f-structures, using the automatically generated LFG resources presented in Chapter 2.

First, I survey the state-of-the-art in natural language generation in this chapter. Section 4.1 gives a broad overview of the literature on sentence realisation and methods for evaluating natural language generation systems. Section 4.2 describes previous research into LFG-based generation and summarises some characteristics of sentence realisation for Chinese. The following two chapters present two distinct approaches to Chinese sentence generation from LFG f-structures. Chapter 5 mi-

grates the LFG approximation and chart-style generater of Cahill and van Genabith (2006) and Hogan et al. (2007) from English to Chinese data and improves on the previous work by applying recent advances in PCFG parsing to PCFG-based generation models. Chapter 6 presents an alternative method to solve the generation problem by pure dependency-based n-gram models, obviating the detour through c-structure and CFG trees in the PCFG-based model.

4.1 Natural Language Generation

Natural Language Generation (NLG) is a subfield of NLP that is concerned with producing natural language expressions from some underlying computer-internal representation of information. The typical architecture in common state-of-the-art NLG systems is a pipeline with the following three stages (Reiter and Dale, 2000):

Text Planning is the process of selecting the information to convey in the output from a knowledge pool, also known as *content determination*; and organising the overall text structure, also known as *discourse planning*.

Sentence Planning is the process of organising the content of each sentence, also known as *sentence aggregation*; deciding the specific words and phrases to express the concepts, also known as *lexicalisation*; and linking pronouns or other types of reference to domain entities, also known as *referring expression generation*.

Surface Realisation is the process of applying grammar rules to produce syntactically, morphologically, and orthographically correct sentences.

Although the three stages are all central to a real-world NLG system, the present thesis concentrates on the component of surface realisation of a single sentence: the main objective of the research presented in Chapter 5 and Chapter 6 is to produce sentences from given LFG f-structures. A surface realisation generation module (on its own) is e.g. an important component in transfer-based MT (Riezler and Maxwell,

2006), in which target sentences are generated from target language f-structures that have been transferred from the original f-structures of the source language.

4.1.1 Surface Realisation

Surface realisation, or sentence generation, is the final stage of natural language generation. The task is to generate a linearly ordered, grammatical string of morphologically inflected words from an abstract semantic or syntactic representation of linguistic content. This process potentially involves several subtasks: (i) determining the linear order of words and phrases; (ii) inserting necessary function words and punctuation; and (iii) performing morphological inflections.

Surface realisers have taken two broad forms, differing with respect to how abstract input representations are mapped to surface forms: via grammar rules or directly.

4.1.1.1 Generation with Grammar

From a knowledge-based NLP perspective, a sentence realiser uses a module encoding knowledge, in most cases in the form of natural language grammar, which defines the relation between natural language utterances and their corresponding meanings. The realiser builds a sentence from a semantic input by applying the grammar rules to construct syntax or derivation trees. The leaf nodes of the tree, read in left-to-right order (by convention), are the words of the generated sentence. In most existing realisers, unification- or constraint-based grammar formalisms are used, e.g., Lexical-Functional Grammar, Head-Driven Phrase Structure Grammar, Combinatory Categorial Grammar, Tree Adjoining Grammar etc.

Symbolic Approaches Using Handcrafted Grammars This is the traditional way to construct a grammar-based sentence realiser. In practice, this means that generation grammars are hand-crafted capturing deep linguistic analyses developed by linguistic experts. Some prominent symbolic systems are listed below:

- FUF/SURGE (Elhadad and Robin, 1996) is a unification-based systemically-oriented grammar of English that uses Functional Unification Formalism as its underlying grammar.

- Penman/KPML (Bateman, 1997) is a sophisticated realiser based on large-scale grammars written within the framework of Systemic-Functional Linguistics. Grammars have been developed using KPML for a variety of languages including English, German, Dutch, Chinese, Spanish, Russian, Bulgarian, and Czech. One of the goals of KPML is multilingual generation that realises a single input in different languages simply by changing the active grammar being used by the system.

- RealPro (Lavoie and Rambow, 1997) is a text generation engine developed by CoGenTex, Inc. based on Meaning-Text Theory. RealPro takes as input a deep syntactic structure and converts it into natural language text by several sets of rules designed for adding function words, specifying the word order, deciding on the appropriate inflections and orthographies.

- Nitrogen (Langkilde and Knight, 1998a) and its successor Halogen (Langkilde, 2002) map from abstract meaning representations to alternative paraphrases for semantic input by hand-written lexical, morphological and keyword-based grammatical rules.

- LinGO/LKB (Carroll et al., 1999) contains the generator used in the LOGON (a Norwegian-to-English MT) system,[1] which operates from meaning representations based on Minimal Recursion Semantics (Copestake et al., 1995) and generates target language realisations in accordance with the LinGo English HPSG Resource Grammar (Flickinger, 2000).

- SUMTIME (Reiter et al., 2003) generates marine weather forecasts for offshore oil rigs from numerical forecast data by manually authored rules and codes, which are informed by corpus analysis.

[1] http://www.emmtee.net/

- XLE (Crouch et al., 2006) provides a generator which takes an LFG f-structure as input and produces all of the strings that, when parsed, could have that f-structure as output. The XLE uses parallel grammars for English, French, German, Norwegian, Japanese, and Urdu, carefully hand-crafted by linguistic experts from the ParGram project (Butt et al., 1999).

The knowledge acquisition bottleneck involved in hand-crafting grammar rules puts traditional, symbolic grammar-based sentence realisers at a disadvantage in that such resources are:

- knowledge-intensive, time-consuming and very expensive to construct;
- language-dependent, domain-specific and hard to adapt to new domains;
- often incapable of dealing with incomplete, incorrect input or linguistic phenomena not covered in the generation grammar.

Statistical and Hybrid Approaches that have been popular in other fields of NLP, are making headway into NLG and sentence realisation in particular. Hybrid approaches use a mix of hand-crafted and automatically derived resources/grammars, statistical approaches use automatically derived resources only. Compared to traditional rule-based symbolic approaches, statistical (and hybrid) approaches promise the advantage of increasing reusability and robustness of NLG systems or components, as well as reducing the painstaking effort of developing deep, wide-coverage grammars. Over the last decade, a number of researchers have presented methods to use statistics and/or grammatical resources derived from a corpus to inform heuristic decisions during what is otherwise symbolic generation:

- FERGUS (Bangalore and Rambow, 2000) is a NLG system involving hybrid techniques. The automatic part of FERGUS is called tree chooser and draws on a stochastic tree model automatically derived from a corpus of XTAG derivations created by transforming the Penn Treebank. The tree chooser takes dependency structures as inputs and outputs supertagged trees using

the probabilistic tree model. The supertagged trees are then linearised by a hand-crafted XTAG English grammar.

- Marciniak and Strube (2004) construe the entire generation process as a number of individual tasks which can be modelled as classification problems. They use the TAG formalism to represent the structure of the generated texts for route directions, and build TAG derivation trees by applying corpus-trained classifiers relying on semantic and contextual features.

- White (2004) integrates statistical n-gram models to prune edges for improving the efficiency of OpenCCG, a chart generator based on a precise, manually developed CCG grammar. White et al. (2007) revise the original CCGbank (Hockenmaier, 2003) by augmenting the lexical categories with semantic representations, and from the converted CCGbank, a broad coverage grammar is automatically extracted and substituted for the previous manually developed grammars used in the OpenCCG realisation.

- Nakanishi et al. (2005) describe probabilistic models for a chart generator based on the Enju grammar, an English HPSG grammar extracted from the Penn Treebank by Miyao et al. (2004). Importing techniques developed for wide-coverage probabilistic HPSG parsing, they apply a log-linear model to pack all alternative derivation trees for a given input into an equivalence class, and apply iterative beam search to reduce the search space during runtime.

- Cahill and van Genabith (2006) and Hogan et al. (2007) present probabilistic surface generation models using wide-coverage LFG approximations automatically extracted from the Penn-II Treebank to determine the most likely f-structure annotated tree (and hence a realisation) given an f-structure (see details in Section 4.2.1).

- Belz (2007) describes a comprehensive approach couched in the format of the Probabilistic Context-Free Representationally Underspecified (pCRU) language framework. The approach includes a base generator creating a genera-

tion space defined by a CRU grammar, and a probabilistic model estimating a probability distribution over the set of CRU grammar rules from a multi-treebank—a treebank containing all derivations licensed by the CRU grammar for an unannotated corpus of example texts. Then the probability distribution is used in one of several ways to drive generation processes, maximising the likelihood either of individual expansions or of entire generation processes. The work of Belz (2007) can be considered as a generalisation of Nakanishi et al. (2005); Cahill and van Genabith (2006); White et al. (2007) etc., which focus on more concrete unification-based grammar formalisms and more specific tasks.

- Zhong and Stent (2005); DeVault et al. (2008) explore the possibility of acquiring probabilistic generation grammars from unannotated data rather than treebanks as in the work described above. They use general purpose tools and resources to parse a set of raw texts into syntactic trees, and augment parse trees with semantic information. They both extract probabilistic tree-adjoining grammars from the annotated parse trees, which are further used to drive the process of sentence realisation. Both report that there is no significant quality difference between sentences generated based on grammars learned from automatically produced parser output trees and hand-corrected or treebank trees.

Generate-and-Select Approach is a NLG methodology that is distinct from one-step grammar-based realisation methods by a clear separation between generation and selection. In this paradigm, either hand-crafted or automatically acquired grammar rules are applied to generate a space of all possible paraphrases on the one hand, and statistical methods (such as word n-grams) are used to select the overall most likely realisation(s) from the space on the other. The attractive aspect of this paradigm is that it partially overcomes the knowledge acquisition bottleneck in NLG by tapping the vast knowledge inherent in large text corpora (for disambiguation), while maintaining the flexibility associated with symbolic systems (and one may still

need a generation grammar). Some traditional hand-crafted surface realisers adopting this methodology have been augmented with a stochastic ranker. By and large, two different probabilistic models have been applied in ranking potential outputs: n-gram language models and log-linear feature models.

N-gram language models introduced statistical approaches in NLG and continue to be popular today. The first significant attempt to incorporate statistical knowledge into surface realisation is Nitrogen (Langkilde and Knight, 1998b), which represents the set of alternative realisations as a word lattice (a state transition diagram with links labelled by words) and selects the best output from the lattice by basic unigram and bigram language models trained on 250 million words of WSJ newspaper text. Langkilde (2000) describes a more efficient ranking algorithm by replacing the word lattice in Nitrogen with a forest representation (a packed set of trees with AND-OR relations), which offers advantages in compactness and the ability to represent syntactic information. In the same way as Nitrogen, FERGUS (Bangalore and Rambow, 2000) generates a word lattice containing all possible realisations by an XTAG grammar and ranks these alternatives in the order of their likelihood by a trigram language model constructed from a 1,000,000 word WSJ corpus. White et al. (2007) rank alternatives by a variety of factored trigram models. Their results show that factored models that integrate word-level n-grams with n-grams over part-of-speech tags and supertags (category labels) provide performance improvements over pure word-level n-grams.

Log-linear (or maximum entropy) feature models are more powerful than conventional n-gram language models in that they incorporate (in principle arbitrary) syntactic and semantic features. Velldal et al. (2004) and Velldal and Oepen (2005) present discriminative maximum entropy models using structural features trained on a small, domain-specific HPSG symmetric treebank[2] comprising 864 sentences constructed using a small hand-crafted HPSG grammar. Their results sug-

[2]Velldal et al. (2004) defined a *symmetric treebank* as a set of pairings of surface forms and corresponding semantics, where (a) each surface form is associated with the sets of alternative analyses; and (b) each semantic representation is paired with the sets of alternate realisations.

gest that the structural model compares favourably against the traditional 4-gram language model trained on all of the British National Corpus containing roughly 100 million words, and in turn that a combined model (with n-grams as a separate feature) outperforms both individual models. Nakanishi et al. (2005) present similar probabilistic models, but using more compact representations of packed feature forests and a wide-coverage HPSG grammar automatically acquired from the much larger Penn-II Treebank. Cahill et al. (2007) implement a log-linear model within the LFG framework and concentrate on a less configurational language, German, using a large hand-crafted LFG grammar.

Comparing n-gram models with log-linear models, log-linear models have the flexibility of integrating multiple overlapping features, including various structural features, without assuming independence among them, and thus usually outperform pure n-gram language models (Velldal and Oepen, 2005; Nakanishi et al., 2005; Cahill et al., 2007). On the other hand, a language model is trained on a raw text corpus that is easily accessible in large quantities, whereas log-linear models incur the overhead of building or annotating a corpus with more linguistic (syntactic or semantic) information, such as the symmetric treebanks used for training the statistical models in Velldal et al. (2004) and Velldal and Oepen (2005).

4.1.1.2 Generation without Grammar

An alternative sentence realisation paradigm is to map from concepts or semantic representations to surface strings *directly* rather than using a full grammar or knowledge base to map between generation input and surface strings. An important advantage of this paradigm, such as using templates,[3] is that they sidestep the complex syntactic structures encoded in large number of grammar rules. A few generation systems following this methodology have been developed, but to the best of my knowledge, to date this approach has been limited to small-scale and specialised applications.

[3]It can, of course, be argued that templates do in fact constitute "cheap" surrogate grammars.

- YAG (McRoy et al., 2000) is a real-time, template-based natural language generator, which extends the traditional template-based approach by allowing templates to embed several types of control expressions, in addition to simple string values. YAG is designed for general purpose applications, but the set of pre-defined templates has to be changed every time for each new domain.

- Ratnaparkhi (2000) construes surface realisation as attribute ordering and lexical choice in generation templates, and presents maximum entropy models to learn the optimal surface realisation for a semantic representation of attribute-value pairs from a corpus of templates, restricted to an air travel domain.

- Oh and Rudnicky (2000) describe a stochastic surface realisation system for spoken dialogue systems in the travel reservations domain, similar to Ratnaparkhi (2000), but use an n-gram language model to generate each utterance.

There have also been several studies concentrating on word or phrase ordering, the primary sub-task of sentence realisation.

- Uchimoto et al. (2000) describe a corpus-based method to acquire the order of modifiers in Japanese clauses. They use a maximum entropy model derived from the Kyoto University dependency treebank to estimate the likelihood of the appropriate order of each pair of modifiers in question, and the order of a clause is determined by all the correctly ordered pairs.

- Ringger et al. (2003) present several statistical models to estimate syntactic constituent order in an unordered syntax tree for French and German. Their experiments show that a particular conditional model incorporating a wide range of syntactic and semantic features and implemented by decision trees performs best.

- Filippova and Strube (2007) extend the work of Uchimoto et al. (2000) and adapt it to learn ordering constituents of a main clause in German. The main difference between the two algorithms is that Filippova and Strube (2007) split

the task into two steps: first, the best candidate for the initial sentence position is chosen by a binary classifier; then the order for the remaining postverbal constituents is determined by a maximum entropy classifier. Results indicate that the two-step method works significantly better than the one without the separation.

4.1.2 Evaluation of NLG Systems

Evaluating NLG systems faces the same problems as those that confront the evaluation of machine translation systems: given a set of automatically generated sentences, how close are they to the human-produced or gold sentences in a given context or application? NLG systems have traditionally been evaluated by human judgements. Human evaluation is informative but time-consuming and expensive. In recent years, inspired by the prevalence of automatic evaluation methods in MT and by the growing demands of modern wide-coverage NLG systems, researchers now often provide automatic corpus-based evaluations of NLG systems.

4.1.2.1 Human-Based Evaluation

In general, there are two types of human-based evaluation methodologies:

Intrinsic Type This involves reading and rating the generated sentences for fluency and adequacy by human subjects, and NLG systems are evaluated by comparing the ratings of their generation output.

Extrinsic Type This involves measuring the impact of different generated texts on task performance, e.g. whether the generated texts help decision making; measuring how much experts post-edit generated texts; and measuring how quickly people can read generated texts.

4.1.2.2 Corpus-Based Evaluation

Human evaluation is the ultimate arbiter of generation quality but is very expensive and time-consuming. Particularly since the advent of the wide-coverage NLG sys-

tems, automatically evaluating NLG systems by comparing the generated texts to a corpus of reference or gold-standard sentences have become widely used in recent years, as it is much cheaper and easier to organise than human evaluation.

In grammar-based generation, there is a seemingly irreconcilable conflict between broad coverage and high accurate outputs. It is usually the case that the rules and features are simultaneously too general to rule out undesirable combinations, and yet too restrictive to allow some combinations that are valid. Accordingly, a comprehensive evaluation has to take the metrics of both accuracy and coverage into account when assessing the quality of realisations.

Coverage is defined as the percentage of input representations for which the sentence generator produces strings.

Accuracy is evaluated by various metrics:

Exact match is the percentage of generated sentences that exactly match a corresponding gold or reference sentence.

N-gram precision and recall metrics such as BLEU (Papineni et al., 2002), NIST (Doddington, 2002) and ROUGE (Lin and Hovy, 2003) are well-established evaluation metrics in MT and summarisation communities. These metrics assess the quality of a generated sentence in terms of the statistics of word n-grams: the more of these n-grams that a realisation shares with the reference sentence(s), the better the realisation is judged to be. The most commonly used metric is the BLEU score, which is computed as a geometric average of n-gram precisions p_n of the generated realisation with respect to the reference, using n-grams up to length N and uniform weights w_n summing to one. Then the average n-gram precision is adjusted by a sentence brevity penalty factor BP, as in Eq. (4.1), where c is the length of the

generated string and r is the length of the reference string.

$$BLEU = \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \times BP \tag{4.1}$$

$$BP = \begin{cases} \exp(1 - r/c) & \text{if } c \leq r \\ 1 & \text{if } c > r \end{cases}$$

$$\text{where} \quad w_n = 1/N$$

String edit distance reflects the average number of insertion (I), deletion (D) and substitution (S) errors between the reference strings (R) and the strings in the test corpus produced by the generator. The NIST Simple String Accuracy (SSA) is one of this type of metrics, which is calculated as Eq. (4.2).

$$SSA = 1 - \frac{I + D + S}{R} \tag{4.2}$$

As the task of generation involves reordering of tokens, the SSA metric may penalise a misplaced token twice, for example, as both a deletion from its expected position and an insertion at a different position. Because of insertions and deletions, the total number of operations may be larger than the number of tokens involved for either reference or output strings, and as a result, the SSA metric may be negative (though it is never greater than 1). To overcome this harsh penalty, Bangalore et al. (2000) revised SSA as Generation String Accuracy (GSA). GSA treats deletion of a token at one location in the string and the insertion of the same token at another location in the string as one single movement error (M). This is in addition to the remaining insertions (I') and deletions (D'), resulting in Eq. (4.3).

$$GSA = 1 - \frac{M + I' + D' + S}{R} \tag{4.3}$$

Many researchers (Stent et al., 2005; Belz and Reiter, 2006) are acutely aware that automatic evaluation metrics are limited, as for instance:

- they tend to have a bias in favour of generators that select on the basis of frequency, and against generators based on purely symbolic rules;

- they tend to award higher ratings to systems which follow corpus frequency, while penalising systems which produce perfectly valid lexical and syntactic variations;
- they are sensitive to the size and make-up of the reference corpus, which is in most cases incapable to cover all surface varieties that express the same meaning.

Nevertheless, BLEU-type automatic evaluation metrics are quick, inexpensive, language-independent, and moreover, have been shown to correlate well with human judgments when comparing statistical NLG systems (Bangalore et al., 2000; Belz and Reiter, 2006), and therefore have become the de facto evaluation standard for statistical realisation approaches. I also adopt corpus-based automatic evaluation methods to estimate the performance of my sentence realisers presented in Chapters 5 and 6.

4.2 LFG-Based Generation for Chinese

4.2.1 Generation in LFG

Work on generation in LFG generally assumes that the generation task is to determine the strings of a language that correspond to a specified f-structure, given a particular grammar (Dalrymple, 2001, pp.429). Based on these assumptions, both theoretical and practical explorations into the problem of generation in LFG have been reported in the literature.

Kaplan and Wedekind (2000) explore the formal properties of generation from f-structures. They prove that given an LFG grammar and a fully specified f-structure, the set of strings that corresponds to the particular f-structure according to the grammar is a context-free language. More recent work in LFG generation has build sentence realisers using symbolic or statistical approaches. ParGram/XLE (Crouch et al., 2006) comes with a fully-fledged unification-based generator, which takes an f-structure as input and generates all possible strings that correspond to that f-structure. The core knowledge bases of XLE are bi-directional LFG grammars

consisting of a large quantity of syntactic rules and lexical entries, hand-crafted by linguists. Cahill et al. (2007) describe a two-stage sentence realiser for German, which produces the generation space by the symbolic XLE generator, using a large hand-crafted grammar for German, and ranks the realisations by a log-linear model similar to that used in Velldal and Oepen (2005) but trained on a symmetric treebank built from the German TIGER Treebank. Cahill and van Genabith (2006) and Hogan et al. (2007) present conditional probabilistic models implemented in chart-style generators for surface realisation from f-structures. Their generators use wide-coverage, probabilistic LFG approximations automatically acquired from the Penn-II treebank (Cahill et al., 2002; Cahill, 2004).

In Cahill and van Genabith (2006), the LFG-based probabilistic generation model defines the conditional probability $P(T|F)$, for each functionally annotated c-structure tree T (whose yield is a surface realisation) given an f-structure F. Among the set of all possible trees $\mathcal{T}(F)$ whose corresponding f-structure is F, the generation model searches for the tree T_{best} that maximises $P(T|F)$:

$$T_{best} = \underset{T \in \mathcal{T}(F)}{\operatorname{argmax}} P(T|F) \tag{4.4}$$

Similar to PCFGs, $P(T|F)$ is decomposed as the product of the probabilities of all the functionally annotated CFG rewriting rules $X \rightarrow \beta$ contributing to the tree T, but in addition to the conditioning on the left-hand side (LHS) non-terminal node X (as in the simple PCFG), each annotated CFG rule is also conditioned on the set of f-structure attributes or features $Feats$ belonging to the f-structure to which the LHS node X is ϕ-linked:

$$P(T|F) \;=\; \prod_{\substack{X \rightarrow \beta \; in \; T \\ Feats = \{a_i | \exists v_i \; (\phi(X) \; a_i) = v_i\}}} P(X \rightarrow \beta | X, Feats) \tag{4.5}$$

For example, the annotated CFG rule $S[\uparrow{=}\downarrow] \rightarrow NP[\uparrow\text{SUBJ}{=}\downarrow] \; VP[\uparrow{=}\downarrow]$ is the expansion of node n_6 which is ϕ-linked to f_3 in Figure 4.1, thus the conditioning factor on the probability of this expansion is the node $S[\uparrow{=}\downarrow]$ and the set of the features

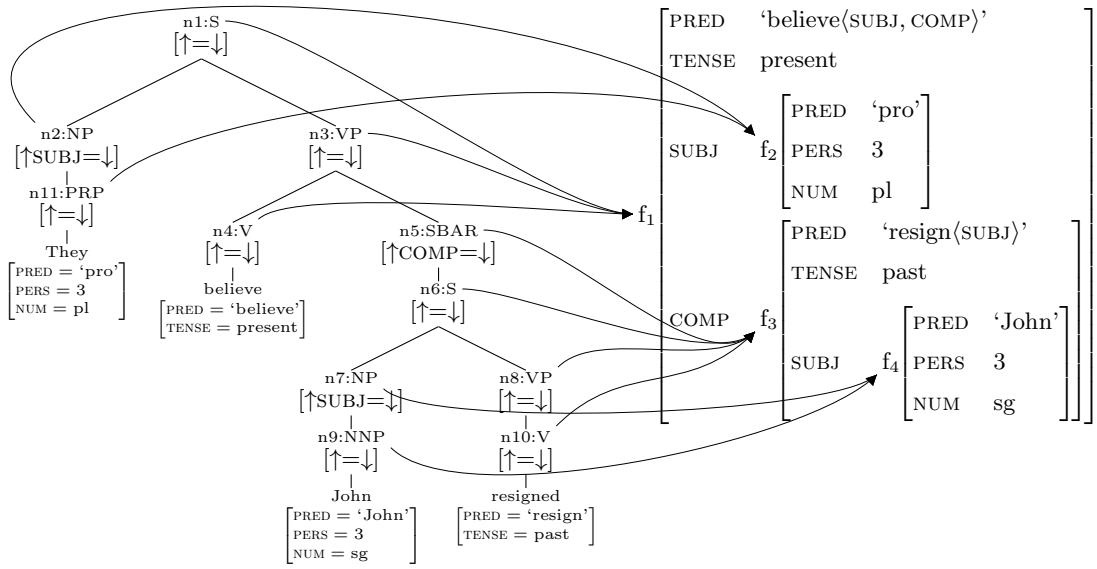


Figure 4.1: C- and f-structures with ϕ links for the sentence *They believe John resigned*

in its ϕ-linked sub-f-structure f_3, i.e. $\{\text{PRED}, \text{TENSE}, \text{SUBJ}\}$.

Based on Cahill and van Genabith (2006), Hogan et al. (2007) present a generation model that improves on the generation accuracy by increasing conditioning context in PCFG style rules. More specifically, the model conditions the f-structure annotated CFG rules on their parent grammatical function, in addition to the local ϕ-linked feature set and the LHS node (Eq. 4.6). For example, the conditioning context of the same rule expanding node n_6 also includes the parent GF of f_3 that is ϕ-linked to n_6, that is COMP.

$$P(T|F) = \prod_{\substack{X \rightarrow \beta \ in \ T \\ Feats = \{a_i | \exists v_i \ (\phi(X) a_i) = v_i\} \\ \exists f \ (f \ \mathbf{GF}) = \phi(X)}} P(X \rightarrow \beta | X, Feats, \mathbf{GF}) \tag{4.6}$$

The generation models described in Cahill and van Genabith (2006) and Hogan et al. (2007) resemble PCFGs in the sense that they use (f-structure annotated) CFG rules extracted from the Penn-II treebank with probabilities, but unlike the standard generative PCFG model, the models have parameters conditioned heavily

on the input f-structure that is non-local to a CFG rule. As in the above example, the conditioning f-structure feature TENSE comes from the lexical annotation on node n_{10} that is not immediately dominated by node n_6, and hence is not present in the rule $S[\uparrow{=}\downarrow] \rightarrow NP[\uparrow\text{SUBJ}{=}\downarrow]\ VP[\uparrow{=}\downarrow]$ expanding the node n_6. In other words, these probabilistic generation models include global conditioning features beyond a generative derivation, therefore they are not standard or even history-based generative PCFG but conditional models.

Nevertheless, the generation methodology proposed by Cahill and van Genabith (2006) is attractive in that it relies on treebank-based automatically acquired grammars and hence is easy to port to new languages. In Chapter 5, I investigate how well the treebank-grammar-based generation methodology migrates from English to Chinese LFG resources, and, in addition, design a sentence realiser based on proper generative PCFG models rather than conditional models. I also investigate how effectively PCFG generation benefits from recent advances in PCFG parsing, such as parent annotation and lexicalisation.

4.2.2 Generation for Chinese

Most work reviewed in the previous sections has been carried out on sentence realisation for English. There has only been a small amount of research into Chinese sentence generation, limited to specific tasks or as part of MT or dialogue systems. Li et al. (1996) present a technique for Chinese sentence generation in the KANT knowledge-based machine translation system,[4] where Chinese sentences are generated directly from interlingua expressions using a unification-based generation formalisation which takes advantages of certain Chinese linguistic features. Liu et al. (2005) describe an algorithm for Chinese sentence generation used in an expert system, where natural language sentences are generated from conceptual graphs by applying simple patterns. Gulila (2005) describes the manual construction of Chinese syntactic, lexical and morphological resources for an off-the-shelf generator originally designed for English to generate Chinese sentences for second language

[4] http://www.lti.cs.cmu.edu/Research/Kant/

learning. Fang et al. (2006) presents a hybrid-template method for generating Chinese simple sentences in a spoken dialogue system. I am not aware of any study on wide-coverage robust statistical sentence generation for Chinese, nor any published work on Chinese generation couched in the framework of LFG.

Contrary to parsing Chinese, which is generally considered to be harder than parsing English due to the tremendous ambiguities caused by the underspecification of Chinese grammar (as regards tokenisation, lack of morphological marking, frequent omission of heads, arguments and modifiers in context etc.), it is usually taken for granted that Chinese generation is easier than generating English. Arguably, this is the case to a limited extent. Unlike English or other languages with rich morphology, performing inflections/declensions is trivial for generating Chinese because of the morphological paucity of the language. As a result of this, surface realisation for Chinese mainly concerns the remaining two subtasks, i.e. determining the word order and inserting function words and punctuation marks. Punctuation is important for clarifying the meaning of sentences; however, as language-specific syntactic elements, punctuation is not conventionally expressed in f-structures which encode more abstract grammatical relations.[5] In the long run, I intend to reproduce punctuation for sentence realisations without punctuation marks by a separate model, but this component is not included in the present thesis.[6] Function words[7] in Chinese serve as a mechanism of reflecting tense, aspect, mood or separating different constituents. For example, the sentence-final particle 吗/*MA* indicates that (1) is an interrogative sentence.

(1) 有 帮助 吗 ?
have help *MA* ?
'Is it of any help?'

The function word 的/*DE* is used as a delimiter between the head noun and its modifier as in (2).

[5]Our Chinese f-structures do present a limited number of punctuation marks, but only restricted to those which represent clause type or connect conjuncts in a coordination.

[6]Note, however, that our generation output is evaluated against gold sentences containing full punctuation, and any missing or incorrect punctuation in generation output is reflected in the evaluation scores.

[7]The term "function words" here is used specifically to denote structural particles, aspectual particles and modal particles. More generally, and outside this thesis, the term also refers to other parts of speech, such as adverbs, prepositions, conjunctions, and so on.

(2) 英文 的 普及度
English *DE* popularisation
'popularisation of English'

The number of function words is limited and most of them occupy particular positions in sentences. For instance, *吗/MA* always appears at the end of a sentence and *的/DE* usually attaches to the modifier of a noun phrase. The limited and regular usage of function words is not difficult to capture by linguistic heuristics.

The primary issue of sentence realisation from Chinese f-structures is to arrange phrases or words in order. As in English, word order in Chinese is fairly rigid and is crucial to establishing the intended reading of sentences. In general, Chinese is an SVO language, nevertheless some varieties are allowed under certain circumstances. For example, all three sentences in example (3) are grammatical and basically have the same meaning, yet achieving somewhat different communicative goals. (3a) is the neutral, unmarked way to express *I bought a book*; (3b) is a topicalised form of the sentence on condition that *书/book* is the current topic of the conversation; (3c) is also acceptable in a situation in which *书/book* is the focus of the sentence or the new information being conveyed by the sentence.

(3) a. 我买 书 了 。
I buy *book* LE .
'I bought a *book*.'

b. 书 我买 了 。
book I buy LE .

c. 我书 买 了 。
I *book* buy LE .

Though the three sentences bear the same basic f-structure with respect to argument functions, they are distinguished by grammaticalised discourse functions TOPIC and FOCUS. In this sense, the relationship between surface realisation and grammatical functions is somewhat direct in Chinese.

Traditionally, in LFG or other unification-based formalisms, generation is regarded as the reverse process of parsing and resolved via application of bi-directional grammar rules. I explore this approach with the PCFG-based chart generation method for Chinese in the next Chapter. However, as has become apparent in parsing, traditional CFG-based syntactic descriptions are probably not the most

adequate formalism for Chinese natural language processing, in as much as ambiguities can interfere with word segmentation, POS tagging and syntactic bracketing. And more importantly, it is the grammatical relations or semantic roles rather than syntactic categories that govern the word order of Chinese sentences. Based on this observation, I develop a novel dependency-based n-gram model for Chinese sentence realisation in Chapter 6, which directly linearises the GFs in f-structures without recourse to an underlying (f-structure-annotated) CFG grammar and CFG-based realisation charts, as in the traditional CFG-based generation models.

Chapter 5

PCFG-Based Chart Generation

5.1 Introduction

Using formal linguistic grammars to generate natural language text from a specified semantic representation is parallel to parsing with these grammars.[1] During parsing, the grammar is used to map from a surface sentence to a representation of the sematic content of that sentence. In surface realisation it goes in the opposite direction, that is, the realiser takes as input a semantic representation that is similar to the output produced by parsers, and produces from this as output a surface sentence that expresses this semantic content. In this sense, the process of surface realisation can be viewed as the inverse of the parsing process. In this chapter, I build a sentence realiser with this inverse parsing approach and reuse the LFG resources automatically extracted from the Penn Chinese Treebank in Section 2.3 as the bi-directional grammar. I adopt the chart generation methodology of Cahill and van Genabith (2006) but: (i) adapt it to Chinese data and automatically acquired wide-coverage Chinese LFG grammars; (ii) implement a sentence generator using proper generative PCFG models rather than the conditional models of Cahill and van Genabith (2006); and (iii) improve on the generation accuracy by breaking down inappropriate independence assumptions in the simple PCFG generation model via

[1] Here "parsing" is used generally for what is more strictly referred to by the term "deep parsing". In contrast with "shallow parsing", "deep parsing" produces a sematic representation in addition to a pure syntactic representation of a sentence.

techniques have been proved successful in PCFG parsing.

This chapter is structured as follows: Section 5.2 describes PCFG-based generation models, moving from the very basic PCFG model to two models incorporating more contextual information. One model uses parent annotation of c-structure categories which was originally designed for PCFG parsing, the other is related and inspired by Hogan et al. (2007), including the parent GF from input f-structures. Section 5.3 describes the chart-style generation algorithm. Section 5.4 gives the experimental results and compares the performance of these various PCFG generation models. Finally, Section 5.5 summarises and outlines some future work. Earlier results of some of the work reported in this chapter have been published in Guo et al. (2008b).

5.2 PCFG-Based Generation Models

5.2.1 The Basic PCFG Model

Viewing generation as the reverse process of parsing, the process of building a sentence from a semantic input is to construct syntax or derivation trees (and their yields) by application of grammar rules. In LFG-based generation, a PCFG generation model assigns a probability to each functionally annotated constituent tree T for a given f-structure F, and the goal of the probabilistic model is to pick the most likely tree T_{best} that maximises the probability $P(T|F)$. By definition, the probability $P(T|F)$ can be rewritten as $P(T,F)/P(F)$. Since we are maximising over all candidate trees for the same f-structure, $P(F)$ will be a constant for each tree, so we can eliminate it leading to:

$$T_{best} = \underset{T \in \mathcal{T}(F)}{\operatorname{argmax}} P(T|F) = \underset{T \in \mathcal{T}(F)}{\operatorname{argmax}} \frac{P(T,F)}{P(F)} = \underset{T \in \mathcal{T}(F)}{\operatorname{argmax}} P(T,F) \tag{5.1}$$

Furthermore, since each constituent tree augmented with consistent functional annotations[2] admits one and only one minimal solution — the smallest f-structure that

[2]Consistent f-structure annotations assign exactly one particular value to each attribute.

satisfies the constraints expressed in the annotations[3] (by means of the constraint solver), viz. $P(F|T)$ is 1. Thus:

$$P(T, F) = P(T)P(F|T) = P(T) \tag{5.2}$$

The final equation for choosing the most likely functionally annotated constituent tree (and its yield) neatly simplifies to choose the functionally annotated constituent tree with the highest probability:

$$T_{best} = \underset{T \in \mathcal{T}(F)}{\operatorname{argmax}} P(T) \tag{5.3}$$

This is identical to the generative or joint PCFG model for parsing. And just like parsing, the probability of the functionally annotated c-structure tree is defined as the product of probabilities of all the n annotated CFG rules of the form $X_i \rightarrow \beta_i$ involved in the derivation:

$$P(T) = \prod_{i}^{n} P(X_i \rightarrow \beta_i | X_i) \tag{5.4}$$

To estimate the rule probabilities $P(X \rightarrow \beta|X)$ of the generation grammar, we follow the method presented in Section 2.3 to annotate the phrase structure/c-structure trees of the CTB5.1 with functional equations that relate c-structure trees to corresponding f-structures. Given the f-structure annotated version of CTB, the probability of each functionally annotated CFG rule can be computed by simple Maximum Likelihood Estimation (MLE):

$$P(X \rightarrow \beta|X) = \frac{Count(X \rightarrow \beta)}{\sum_{\gamma} Count(X \rightarrow \gamma)} = \frac{Count(X \rightarrow \beta)}{Count(X)} \tag{5.5}$$

5.2.2 Models with Increased Structural Sensitivity

PCFGs are a natural starting point for parsing and generation. Unfortunately, research suggests that PCFGs are poor models of language in several respects:

[3]We stipulate that disjunctions are not included in functional annotations in our treebank-based annotation regime.

poor independence assumptions: CFG rules impose independence assumptions on probabilities, resulting in poor modelling of structural dependencies across the syntax tree.

lack of lexical information: CFG rules do not model syntactic facts about specific words, leading to problems with ambiguities of prepositional phrase attachments and coordination structures, and so forth.

Because of these problems, most current probabilistic parsing models use some augmented version of PCFGs, such as tree transformations (Johnson, 1998) and lexicalisation (Charniak, 1997; Collins, 1999), which have shown significant improvements over simple PCFGs. However, it is interesting to note that there has been a lot less research on this subject for sentence generation — a process generally regarded as the reverse of parsing. In this section I investigate the effect of increasing the context sensitivity of PCFG models on the performance of PCFG-based generation. Another approach to improving simple PCFGs is to include lexical dependencies, which is not presented in this thesis, but is a promising direction to be explored in the future.

5.2.2.1 Annotation with Parent Category

A simple PCFG embodies independence assumptions about the distribution of words and phrases, viz. the probability of each production is independent of the context beyond the local tree from which the production is extracted. Unfortunately this "context-free" independence assumption results in poor probability estimates, as it is far too strong for natural language grammars. Consider the expansion of NP nodes as temporal nouns: $NP \rightarrow NT$. As indicated by statistics for NPs in the CTB5.1 training set (Table 5.1), an NP in Chinese is more likely to be a temporal noun if it functions as an adverbial modifier, even though an NP is much less likely to be a temporal noun in many other situations (such as when it functions as a subject or attribute modifier). However, the basic PCFG model does not represent this contextual difference in the probabilities. As exemplified by tree (a) in Figure 5.1,

	Temporal Nouns	Other Nouns
subject	67(0.2%)	32,811(99.8%)
attribute modifier	858(3.2%)	25,649(96.8%)
adverbial modifier	**2,595(61.2%)**	1,647(38.8%)

Table 5.1: Distribution of NPs in different context in CTB5.1

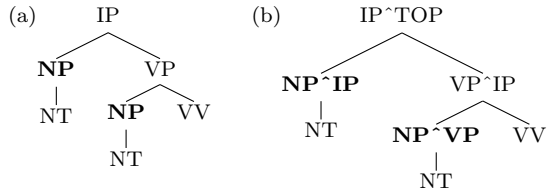


Figure 5.1: Trees before and after "parent annotation"

in the simple PCFG model the upper and lower NPs have the same expansions and these expansions have the same distribution.

To relax the independence assumptions implicit in a PCFG, Johnson (1998) proposed a node relabelling method augmenting the node's label with the category of the node's parent so as to encode additional information about the context in which the node appears. For the example of NP expansions, the upper NP bearing a subject function is distinguished from the lower NP modifying the matrix verb by different parent annotations IP and VP as in tree (b). Johnson reported that applying this simple parent annotation transformation to the Penn-II treebank trees improved the PCFG estimated from the trees, and yielded an increase in both precision and recall of a parser based on this PCFG by around 8%. Although this kind of tree transformation has been proved to have a remarkable effect on the performance of treebank PCFG-based parsers, as yet I am not aware of any result for applying this technique to wide-coverage PCFG-based generation.

Following Johnson's parent annotation, I transform the f-structure annotated treebank trees by appending the phrasal category of each parent node onto the label of all of its nonterminal children, as in Figure 5.2.[4] Notice that the f-structure annotated CFG rules mapping between a sequence of words and the corresponding f-structure, consist of syntactic categories and f-structure equations. Compared to simple CFG rules used in parsing, they are more fine-grained in that the same

[4]A dummy category label TOP is appended to root nodes.

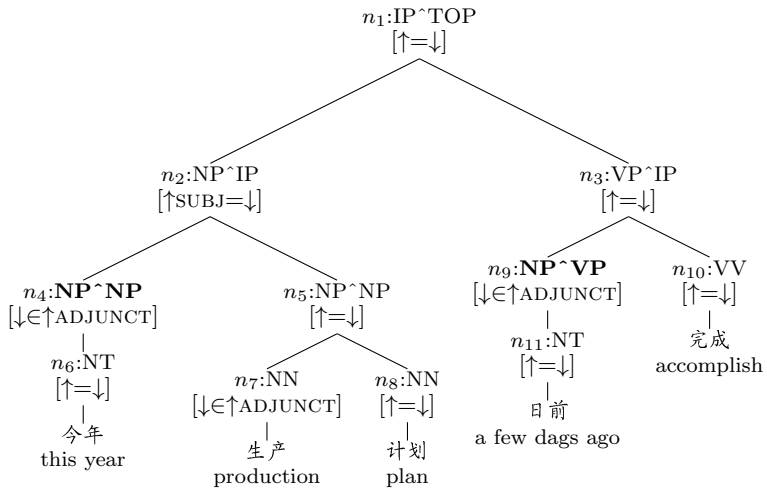


Figure 5.2: Annotation with parent category on the functionally-annotated tree for the sentence *今年生产计划日前完成/The production plan for this year has been accomplished a few days ago*

expansions are distinguished by the functional equations to a limited degree, e.g. $NP[\uparrow\text{SUBJ}=\downarrow] \rightarrow NT[\uparrow=\downarrow]$ and $NP[\downarrow\in\uparrow\text{ADJUNCT}] \rightarrow NT[\uparrow=\downarrow]$ become two different expansions (which otherwise are the same in normal PCFGs without function equations). Nevertheless, the "structural sensitivity" encoded in the syntactic category of the parent node is still capable to discriminate among particular expansions, for example, the adverbial modifier *NP* node n_9 that is attached to a *VP* label differs from the attribute modifier *NP* node n_4 attached to an *NP* label.

The annotation with parent category has an effect equivalent to splitting non-terminal nodes to reflect different contexts in which they occur. In the basic PCFG, each production $X \rightarrow \beta$ has probability $P(X \rightarrow \beta|X)$; in the parent category annotated PCFG (PC-PCFG), the parameters are extended to condition on the additional syntactic category of X's parent $Parent(X)$. Formally, the PC-PCFG generation model is defined as:

$$P(T|F) = \prod_{i}^{n} P(X_i \rightarrow \beta_i | X_i, \boldsymbol{Parent(X_i)}) \tag{5.6}$$

5.2.2.2 Annotation with Parent GF

Johnson's parent annotation transformation provides an effective way to systematically encode contextual information in the structure of individual node labels. In principle, any contextual information could be included if it is local to the relabelled node. Arbitrary contextual information, however, would quickly lead to unacceptable sparseness under MLE. In PCFG parsing, a number of strategies to enrich or split nodes' labels using different contextual information have been reported in Klein and Manning (2003), Levy and Manning (2003), Petrov et al. (2006) etc.

In LFG-based generation, Hogan et al. (2007) present a conditional history-based probabilistic model to overcome some of the inappropriate independence assumptions in the original generation model of Cahill and van Genabith (2006). The proposal of Hogan et al. (2007) increases the conditioning context by including the parent grammatical function of the given f-structure in addition to the local f-structure feature set when predicting grammar rule expansions. Including the parent GF as a conditioning feature has the effect of making the choice of generation rules sensitive to the functional context of the given f-structure. For example, generation rules for pronouns are distinguished between subject and object contexts by the SUBJ and OBJ parent GFs of the ϕ-linked f-structures. This helps the model to correctly generate a nominative pronoun in the subject position and an objective pronoun in the object position for English, as in the sentence *she hired her*. Hogan et al. (2007) showed that including the f-structure parent GF significantly improved generation accuracy over the model of Cahill and van Genabith (2006). Tested on Section 23 of the English Penn-II Treebank, the history-based probabilistic model improved BLEU score from 0.6652 to 0.6724 and SSA score from 0.6869 to 0.6989.

Even though the generation model presented in Hogan et al. (2007) is a conditional probabilistic model, the additional conditioning feature of the parent f-structure GF is related to each c-structure node by the piecewise correspondence ϕ, and thus can be incorporated in a generative PCFG model with more context sensitivity while remaining in the CFG paradigm. Analogous to the annotation with the parent phrasal category, Figure 5.3 exemplifies a CFG tree annotated with the

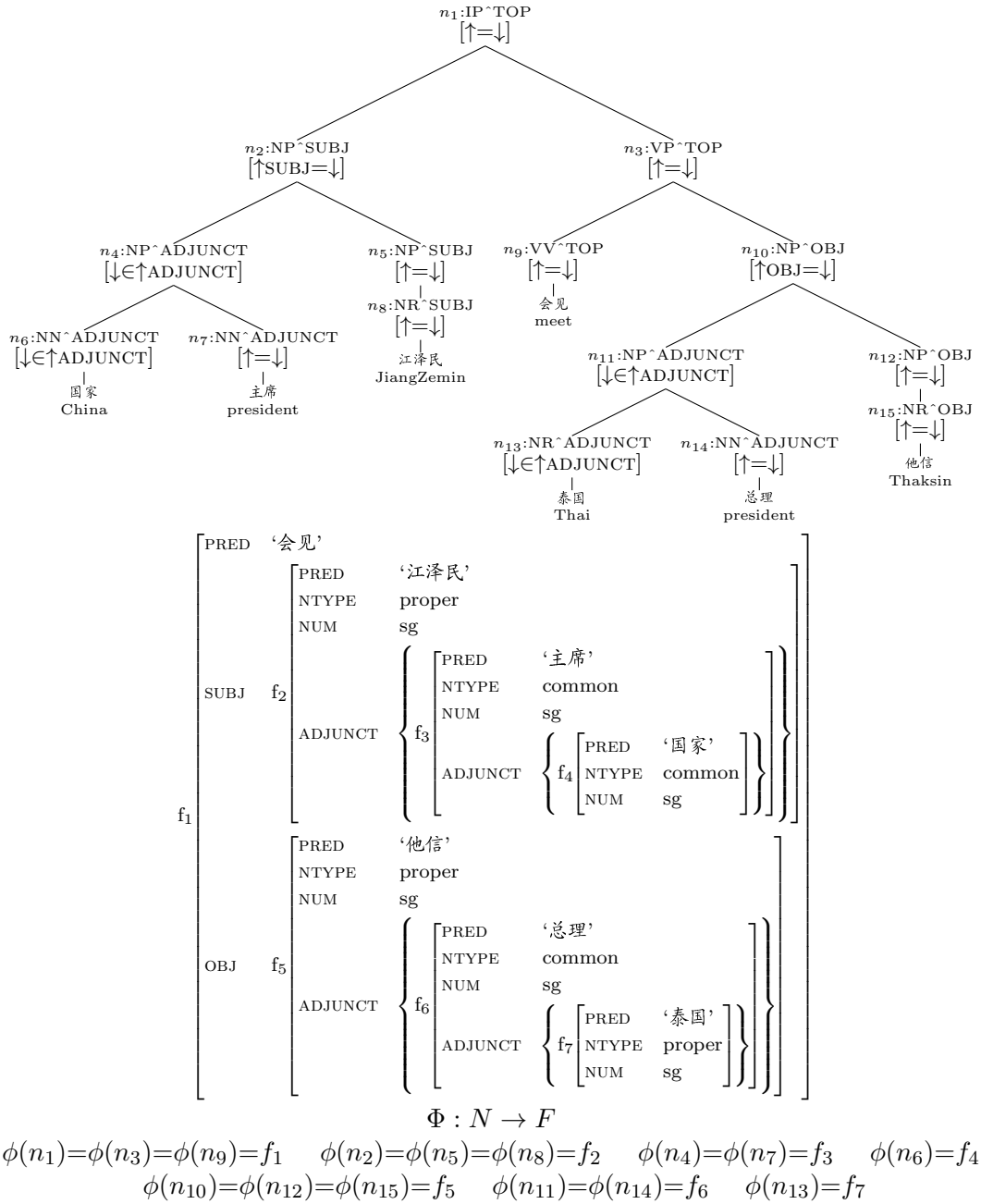


Figure 5.3: Annotation with parent GF on the functionally-annotated tree for the sentence *国家主席江泽民会见泰国总理他信/Chinese President JiangZemin met with Thai president Thaksin*

parent grammatical function of the ϕ-linked f-structure.[5] The parent GF annotation subverts some poor independence assumptions in the basic PCFG (even annotated with f-structure equations). For example the NP nodes n_5 and n_{12} have the same expansion $NP[\uparrow{=}\downarrow]{\rightarrow}NR[\uparrow{=}\downarrow]$ in a simple PCFG, whereas they are distinguished by the parent annotation with SUBJ and OBJ GFs. The parent GF annotated PCFG (PF-PCFG) model can be defined in the same form as the parent category annotated PCFG (PC-PCFG) model, differing in the value of the conditioning context:

$$P(T|F) = \prod_{i}^{n} P(X_i \rightarrow \beta_i | X_i, \boldsymbol{Parent\ GF(\Phi(X_i))}) \tag{5.7}$$

To some extent the annotations of parent syntactic category and grammatical function have the same effect of discriminating between different contexts. For example, to distinguish a subject NP node from an object NP node, the PC-PCFG model splits NP nodes by appending the parent label IP to a subject NP, while appending the parent VP to an object NP. The PF-PCFG model has the same ability to distinguish NP nodes by explicitly annotating them with the ϕ-linked SUBJ or OBJ function. Nevertheless the PF-PCFG and PC-PCFG models produce different probability distributions because the correspondence between c-structure nodes and f-structure units is a many-to-one mapping that results in different mother-daughter relationships between a c-structure and the corresponding f-structure. On one hand c-structure nodes annotated with the same parent phrasal categories can be distinguished by different parent GFs in the PF-PCFG model, as for example the NR nodes n_8 and n_{15} in Figure 5.3: they are both dominated by an NP node, but n_8 is ϕ-linked to f_2 which is the subject of its parent f-structure f_1, while n_{15} is linked to f_5 which is the object of f_1. On the other hand nodes annotated with the same parent GFs can also be distinguished by different parent syntactic labels in the PC-PCFG model, as for example the NP nodes n_4 and n_9 in Figure 5.2, which both relate to an ADJUNCT f-structure, but the nominal modifier n_4 is a child of an NP node while the adverbial modifier n_9 is a child of a VP node. It would therefore

[5]The parent grammatical function of the outermost f-structure is assumed to be a dummy GF TOP.

be interesting to investigate the effectiveness of the parent annotation with syntactic categories and grammatical functions in improving simple PCFGs in the field of PCFG-based generation and to compare the performance of the PC-PCFG and PF-PCFG generation models.

5.3 Chart Generator

5.3.1 Generation Algorithm

Cahill and van Genabith (2006) implemented their probabilistic sentence generator in a chart-based architecture similar to that introduced by Kay (1996). Kay's original chart algorithm, however, indexed edges with semantic variables in order to efficiently find edges that can interact with each other given a relatively flat semantic input. By contrast, LFG f-structures are somewhat more hierarchical representations which provide a schema to build a chart-style data structure by indexing each (sub-)f-structure with a (sub-)chart. Although a conditional probabilistic model underlies the generator of Cahill and van Genabith (2006), the mechanism of the f-structure-indexed chart generator also suits standard generative probabilistic models well. For this thesis, I reimplement the chart generator for the generative PCFG generation models based on this insight. The generator recursively generates the most probable CFG trees for each level of the given f-structure in a bottom-up manner. At each sub-chart, a CKY-like algorithm (with binarised grammars) produces derivations as follows:

1. Generating lexical edges[6] from the local grammatical function PRED.

2. Generating lexical edges from the atomic-valued features representing auxiliary, mood or aspect etc. that are realised by function words. Table 5.2 lists all the function word features in the Chinese f-structures.

3. Applying unary rules and binary rules to generate new edges until no more new edges can be generated in the current local chart.

[6]All phrases of the same category that cover the same semantic relation (or lexicon) are equivalent for the purpose of constructing larger phrases and called "edges" in charts.

4. Propagating edges compatible with the parent GF of the local f-structure to the upper-level chart.

Features	Description	Example
de	的/DE	不小/not small *的/DE* 差距/distance
di	地/DI	神秘/mysteriously *地/DE* 走出来/come out
dr	得/DR	用/use *得/DR* 极佳/very well
ASPECT	aspectual particle	获得/won *了/LE* 胜利/victory
MOOD	modal particles	说/say 什么/what *呢/NE*
MSP	other particles	随/follow 之/it *而/to* 去/go
VNV-FORM	verb duplication	能/can *不/not* 能/can 帮忙/help

Table 5.2: Atomic-valued features for function words

Figure 5.4 shows the chart generating the sentence *江泽民会见泰国总理/Jiang Zemin met with Thai president* for the given f-structure. The chart is composed of four sub-charts Ci, each of which is indexed by a sub-f-structure f_i in the overall f-structure. The generation process starts from the inner-most sub-f-structure f_4. For lexical edges, the algorithm checks the set of atomic/lexical attributes and corresponding values at the sub-f-structure, which in this case is {PRED='泰国',NTYPE=proper,NUM=sg}. All lexical rules matching this particular feature set are applied. In this example two lexical edges 4-1 and 4-2 with the same yield *泰国/Thai* but different categories[7] are added into the sub-chart $C4$. Then unary productions are applied if the RHS of the unary production matches the LHS of an edge currently in the sub-chart. If two or more generation rules with equal LHS categories can be applied, only the edge generated with higher probability is added into the chart for efficiency purposes in the implementation. In this example, two new edges 4-3 and 4-4 are generated from the existing edge 4-1. At $C4$ there are no binary rules that can be applied. At this stage, it is not possible to add any more edges, therefore the algorithm propagates edges in $C4$ that are compatible with the parent GF of $f4$, ADJUNCT in this case. Edges 4-2, 4-4 whose functional equation on the LHS category is [↑ADJUNCT=↓] are propagated to sub-chart $C3$, which is indexed with the upper-level f-structure f_3, for consideration in the next iteration.

[7]The f-structure annotated CFG rules use complex categories including two parts: traditional syntactic categories and functional equations. Two categories having the same syntactic tag but different functional equations are hence different categories.

$$
\begin{bmatrix}
\text{PRED} & \text{‘会见’} & \\
\text{SUBJ} & f_2 \begin{bmatrix} \text{PRED} & \text{‘江泽民’} \\ \text{NTYPE} & \text{proper} \\ \text{NUM} & \text{sg} \end{bmatrix} & \\
\text{OBJ} & f_3 \begin{bmatrix} \text{PRED} & \text{‘总理’} \\ \text{NTYPE} & \text{common} \\ \text{NUM} & \text{sg} \\ \text{ADJUNCT} & \left\{ f_4 \begin{bmatrix} \text{PRED} & \text{‘泰国’} \\ \text{NTYPE} & \text{proper} \\ \text{NUM} & \text{sg} \end{bmatrix} \right\} \end{bmatrix} &
\end{bmatrix}_{f_1}
$$

	Edges			
	No.	Grammar Rule	Realisation	Sources
C4	4-1	NR[↑=↓]→{PRED=‘泰国’,NTYPE=proper,NUM=sg}	泰国	lexicon
	4-2	NR[↑ADJUNCT=↓]→{PRED=‘泰国’,NTYPE=proper,NUM=sg}	泰国	lexicon
	4-3	NP[↑=↓]→NR[↑=↓]	泰国	4-1
	4-4	NP[↑ADJUNCT=↓]→NR[↑=↓]	泰国	4-1
C3	3-1	NN[↑=↓]→{PRED=‘总理’,NTYPE=common,NUM=sg}	总理	lexicon
	3-2	NP[↑=↓]→NN[↑=↓]	总理	3-1
	3-3	NP[↑OBJ=↓]→NR[↑ADJUNCT=↓] NN[↑=↓]	泰国总理	4-2,3-1
		.....	.....	
C2	2-1	NR[↑=↓]→{PRED=‘江泽民’,NTYPE=proper,NUM=sg}	江泽民	lexicon
	2-2	NP[↑=↓]→NR[↑=↓]	江泽民	2-1
	2-3	NP[↑SUBJ=↓]→NR[↑=↓]	江泽民	2-1
		.....	.....	
C1	1-1	VV[↑=↓]→{PRED=‘会见’}	会见	lexicon
	1-2	VP[↑=↓]→VV[↑=↓] NP[↑OBJ=↓]	会见泰国总理	1-1,3-3
	1-3	IP[↑=↓]→NP[↑SUBJ=↓] VP[↑=↓]	江泽民会见泰国总理	2-3,1-2
	1-4	TOP→IP[↑=↓]	江泽民会见泰国总理	1-4
		.....	.....	

Figure 5.4: The chart for the given f-structure of the sentence *江泽民会见泰国总理/JiangZemin met with Thai president*

Sub-chart $C3$ is constructed in a similar fashion. First, lexical edges like 3-1 with yield *总理/president* are added. Next, edges such as 3-2 are generated by unary rules. The edges propagated from the subsidiary chart $C4$ make it possible to apply binary rules to combine them with the new edges generated in $C3$, which results in the new edge 3-3 generating the string *泰国总理/Thai president*. The process continues until it reaches the outmost level of the f-structure f_1, and no more rules can be applied to the existing edges in $C1$. At this stage, the algorithm searches for the most probable edge with TOP as its LHS category and returns the yield of this edge as the final output. If there is no edge with the LHS label TOP in the chart $C1$, the generator only produces a partial output or fails.

Different from parsing where binary rules can only be applied to adjacent edges,

in generation, combinations must be considered between all edges in the current local chart as there is no constraint on string positions, which easily leads to a combinatorial explosion. To address this problem in chart generation, in my implementation I made the following provisions:

- Indexing each sub-f-structure with a sub-chart (as in Cahill and van Genabith (2006)), which bears a broad similarity to indexing edges with semantic variables proposed in Kay (1996). In the scheme of Kay (1996), only edges sharing the same index can interact, and in my implementation, edges can only be combined with those in the same sub-chart. For example, the edges in the sub-chart $C4$ cannot be combined with the edges generated in $C2$, because the two sub-charts are associated with f_4 and f_2, respectively, in the overall f-structure, which have no direct semantic relation between each other. In addition, the grammatical function of the local-f-structure has the effect of preventing incompatible edges from being generated in the indexed sub-chart. For instance, an edge $NP[\uparrow\text{SUBJ}{=}\downarrow]$ with yield *泰国总理/Thai president* can possibly be generated by combining the edges 4-2 and 3-1 via the rule $NP[\uparrow\text{SUBJ}{=}\downarrow] \rightarrow NR[\uparrow\text{ADJUNCT}{=}\downarrow]\ NN[\uparrow{=}\downarrow]$, however, this edge would not be added into the sub-chart $C3$, as its functional equation $[\uparrow\text{SUBJ}{=}\downarrow]$ clashes with the parent GF OBJ of f_3.

- Associating each edge with a bit vector for words to show which of those words the edge covers. Combinations only occur between pairs of edges whose bit vectors have empty intersections, indicating that they do not cover overlapping sets of words. For example, edges 3-2 and 3-3 can not be combined because they contain the same bit element for the word *总理/president*.

- Prohibiting proliferation of grammatically correct, but unusable sub-phrases, which is a particular, identifiable source of the exponential complexity of the chart generator. A well known example is the sentence addressed in Kay (1996) *The tall young Polish athlete ran fast.* We need to guarantee that only the complete noun phrase *the tall young Polish athlete* can be combined

with the verb phrase *ran fast* for the final realisation. However, all partial sub-phrases like *tall athlete*, *young athlete*, *young Polish athlete* and so on, can (in principle) be generated in a chart generator and have the possibility to combine with the rest of the input to construct grammatically correct but incomplete phrases or sentences. To partially solve the problem, my generation algorithm removes subsumed edges during the process of propagation, if the yields of the edges being propagated upwards to the next level sub-chart are subsumed by an edge already in that sub-chart.[8] Similar to Kay (1996)'s strategy of internal indices, this algorithm does not prevent the generation of an exponential number of variants of phrases containing modifiers, but it limits proliferation of ill effects, by allowing only the most probable tree with the longest yield to be propagated upwards and be considered in the next iteration.

5.3.2 Lexical Smoothing

In the PCFG-based chart generator, the number of sentences which can be completely generated is impacted on by the coverage of lexical rules and phrasal grammar rules. I run experiments to assess the extent of the coverage of grammar rules and also the impact of grammar coverage on generation accuracy by examining rule frequencies in training data in Section 5.4.3. Another important factor in generation coverage, similar to the fundamental problem of unknown words in PCFG parsing, is unknown lexical features. In LFG-based generation, the given f-structure encodes the surface form of each lemma in a particular set of lexical features. Lexical rules/entries in the form of *POS*→*Lexeme*[*Lexical Features*] associate the lexical feature sets (and corresponding lexemes) with possible POS tags learned from training data. In the generation chart, phrasal grammar rules are applied to the POS tag sequence generated by the lexical rules. The number of lexical features is potentially unlimited (to encode lemmas for open word classes) and hence can not be covered by

[8]The algorithm of Cahill and van Genabith (2006) adopted a similar strategy in which any edges subsumed by an edge in the same sub-chart would not be generated. This strategy gives the subsumed edges no possibility to interact with other edges for further combination and leads to final failure in some cases.

a training set with limited size. For dealing with unknown lexical features never seen in training data, lexical smoothing is necessary to predict potential POS categories on which CFG derivations can be further conducted.

Cahill and van Genabith (2006) propose a smoothing method to generalise particular lexical features as lexical macros by removing the specific predicate lemma. For example, the lexeme *总理/president* is represented as a lexical feature set {↑PRED='总理',↑NTYPE=common,↑NUM=sg}, the corresponding abstract lexical macro is {↑PRED=$LEMMA,↑NTYPE=common,↑NUM=sg}, which generally associates with common nouns NN in the CTB. The generator of Cahill and van Genabith (2006) extracts lexical macros from the lexical features that occur only once in the training set and uses these lexical macros to create approximating lexical rules for tagging unknown lexical features. In their conditional generation model, the probability of a lexical rule is estimated as Eq. 5.8, where t is a potential POS tag, w is a surface word and f is the corresponding lexical feature.

$$P(t \rightarrow w, f|f) = \frac{count(t \rightarrow w, f)}{count(f)} \tag{5.8}$$

As approximating lexical rules acquired from lexical macros is not as accurate as real lexical rules seen during training, in their generation model, the probability of lexical rules for unknown lexical features is penalised by multiplying by a very small constant. That means that lexical rules seen during training have a much higher probability than lexical rules added during the smoothing phase.

In my PCFG-based generation models, I adopt the same smoothing methodology to generalise lexical features as abstract lexical macros. However the lexical smoothing of Cahill and van Genabith (2006) is applied to a conditional generation model, and moreover, simply multiplying the approximating rules by a smoothing weight breaches the property of MLE in a proper probabilistic model (as no discounting is carried out and extra probability mass is added for the lexical features from which lexical macros are abstracted). For these reasons, I modify the smoothing method to estimate the probability of lexical rules in the same way as other nonterminal

rules in the standard generative PCFG model:

$$P(t \rightarrow w, f|t) = \frac{count(t \rightarrow w, f)}{count(t)} \tag{5.9}$$

According to the assumption that unknown words have a probability distribution similar to hapax legomenon (Baayen and Sproat, 1996), for each lexical rule occurring only once in training data, an approximating lexical rule is generated by replacing the RHS particular lexical feature set with the corresponding more general lexical macro. All approximating rules are added to the set of grammar rules extracted from the training data and probabilities of all rules are computed by MLE. In generation, if the given f-structure includes a set of lexical features that has never been seen in training, the lexical feature set is replaced with the abstract lexical macro, and all approximating lexical rules whose RHS features match the lexical macro will be applied.

As Chinese has very little morphology, it is a common phenomenon that words in Chinese function as different POS categories but have the same word form and the same set of lexical features. For example, the lexical entry 如何{↑PRED='如何'} is associated with AD, DT, JJ, VA and VV, 5 different POS tags in the CTB. It is quite likely that the lexical rules learned from training data can not predict the appropriate POS for a set of lexical features occurring in a particular context of the given f-structure, even though the feature set has been seen associated with other POS tags in training. Therefore, lexical smoothing is necessary not only for unknown lexical features but also known lexical features (with potentially unknown POS tags).

5.4 Experiments and Results

5.4.1 Experimental Data

Experiments are conducted on the CTB5.1 which is split into training, development and test sets as described in Section 2.5.1. For developing the PCFG generation grammar, first the treebank trees of the CTB5.1 training set are automatically as-

sociated with f-structure equations by the annotation-based algorithm presented in Section 2.3. Then functionally annotated CFG rules are extracted from the annotated version of the treebank. Table 5.3 shows the number of CFG rules obtained from the training set by the three generation models: the basic PCFG, the PCFG with annotation of parent grammatical function (PF-PCFG) and the PCFG with annotation of parent syntactic category (PC-PCFG).

Model	Rules
PCFG	8,873
PF-PCFG	10,757
PC-PCFG	11,988

Table 5.3: Number of different types of PCFG rules in the training set

To avoid extraordinarily long sentences which could take a long time to be generated, sentences consisting of more than 40 words are excluded from test and development data, which results in a development set comprising 1,226 sentences (77.01% of the whole CTB5.1 development set) and a test set comprising 1,304 sentences (75.90% of the whole CTB5.1 test set). The original CTB trees of the test and development sets then were automatically translated into f-structures by the annotation-based acquisition algorithm, as input to the generator.

Punctuation is not presented in canonical f-structures except that some special punctuation marks are recognised as conjunctions in my Chinese LFG analysis of coordination constructions. However, punctuation needs to be generated in the final sentence realisations. In future work, I intend to handle punctuation by a separate component during post-processing. In the experiments reported here, I record all punctuation marks in the f-structure representations as a special GF PUNC, and generate them in the same way as regular GFs.

The input to the generator are unordered f-structures, which do not contain any string position information. But, due to the particulars of the automatic f-structure annotation algorithm, the order of sub-f-structures in set-valued GFs, such as ADJUNCT & COORD, happens to correspond to their surface order. To avoid unfairly inflating evaluation results, I lexically reorder the GFs in each local f-structure of the development and test input before the generation process. This

resembles the "permute, no dir" type experiment in Langkilde (2002).

5.4.2 Comparing Conditional and Generative Models

The probabilistic generation models presented in Cahill and van Genabith (2006) and Hogan et al. (2007) are conditional models in that they define probabilities of f-structure annotated productions conditional directly on the sets of the input f-structure features/attributes. By contrast, the probabilistic models presented in this thesis are generative (or joint) models based on derivation of standard or augmented PCFGs.

To compare the performance of my generative PCFG models to the probabilistic models conditioned on input f-structure features, I reimplement the conditional models of Cahill and van Genabith (2006) and Hogan et al. (2007) on the CTB data and carry out experiments with the standard PCFG, the PF-PCFG and the PC-PCFG models. To eliminate the effects of unknown lexical features, I extract lexical rules from all treebank trees of CTB5.1 including the test and development set, so lexical smoothing is not relevant in this experiment.

The generation models are evaluated against the raw text of the testing data in terms of accuracy and coverage. Following Langkilde (2002) and other work on wide-coverage, general-purpose generators, I adopt BLEU score (Papineni et al., 2002), average NIST simple string accuracy (SSA) and percentage of exactly matched sentences for accuracy evaluation. For coverage evaluation, I measure the percentage of input f-structures that generate a sentence. To measure whether the difference between the accuracy scores of two generation models is significant or only due to chance, I employ statistical significance tests. To measure the significance of an improvement in the BLEU score, I use FastMtEval,[9] a bootstrap resampling method which is popular for machine translation evaluations. For SSA scores, I calculate the statistical significance by applying a paired student t-test on the mean difference of the SSA scores. As incompleteness of realisations has a negative impact on the

[9]Scripts for the bootstrapping evaluation of confidence intervals and statistical significance testing are available for download at the author's homepage: http://www.computing.dcu.ie/ nstroppa/index.php?page=softwares&lang=en

BLEU and SSA scores, significance tests comparing two models are conducted only on the intersection of complete sentences which are generated by both models.

Complete Sentence	**Coverage**	**ExMatch**	**BLEU**	**SSA**
Cahill (2006)	87.19%	24.13%	0.7040	0.6682
Hogan (2007)	84.26%	24.39%	0.7103	0.6746
			>Cahill	∼Cahill
PCFG	**98.69%**	23.14%	0.7050	0.6644
			∼Cahill	∼Cahill
PF-PCFG	97.06%	24.03%	0.7142	0.6708
			∼Hogan	∼Hogan
PC-PCFG	97.55%	**24.67%**	**0.7206**	**0.6840**
			≫Hogan	≫Hogan

Table 5.4: Results for completely generated sentences on development data

Table 5.4 gives the comparison results of the five generation models evaluating against the subset of completely generated sentences for f-structures of the development set. In the table, $\gg$ means statistical significance at the level of p=0.005, $>$ means significance at p=0.05 and $\sim$ means the difference is not significant. With regard to coverage, the conditioning factor contributed by f-structure features leads to relatively low coverage for the two conditional generation models. By contrast, the three generative PCFG models boost the number of completely generated sentences by more than 10%. With regard to accuracy, the conditioning f-structure features change the probability distribution over the CFG rules, however they do not result in higher accuracy compared to the generative PCFG models. Specifically, the model of Cahill and van Genabith (2006) performs about the same as the simple PCFG model, while the model of Hogan et al. (2007) which also includes the parent GF as a conditioning feature performs at about the same level as the corresponding PF-PCFG model, but both conditional generation models perform significantly worse than the generative PC-PCFG model. Roughly speaking, three major reasons account for this fact: (i) the generation grammar rules employed by the PCFG models are not conventional CFG rules, but CFG rules annotated with grammatical functions, hence the information contributed by f-structure features is already contained in the annotated CFG rules to some extent; (ii) the implementa-

tion of the chart-style generator associates a sub-chart with each sub-f-structure of the generation input f-structure, and this set-up prevents rules incompatible with the input f-structure to be applied; (iii) the two generation models that condition directly on input f-structure features suffer from severe data sparseness in generation grammar rule counts and overfitting under MLE.

Another observation from the results presented in Table 5.4 is the performance of the three generative PCFG models. The two models that extend the conditioning context of CFG productions to parent annotations have slightly lower generation coverage than the basic PCFG model. However, as far as generation accuracy is concerned, the PC-PCFG model that includes the phrasal category of the parent node outperforms the other two PCFG models. The PF-PCFG model that includes the grammatical function of the f-structure parent is also better than the simple PCFG model, with a significant improvement in the BLUE score and an observable but not significant improvement in the SSA score.

All Sentence	**Coverage**	**ExMatch**	**BLEU**	**SSA**
Cahill (2006)	100%	21.04%	0.6624	0.6403
Hogan (2007)	100%	20.55%	0.6609	0.6410
PCFG	100%	22.84%	0.7034	0.6628
PF-PCFG	100%	23.33%	0.7091	0.6671
PC-PCFG	100%	**24.06%**	**0.7171**	**0.6796**

Table 5.5: Results for all sentences on development data

Table 5.5 lists results evaluating against all (complete and partial) sentences generated from the input f-structures. These results are a natural outcome of Table 5.4. As the conditioning f-structure features do not improve the accuracy but reduce the generation coverage, the generative PCFG models show substantially better overall performance than the conditional generation models. And again, the syntactic category parent annotation model achieves the best results among all the PCFG-based generation models.

5.4.3 Impact of Rule Frequencies

The generation grammar used in the PCFG-based models is a bi-directional, large-scale grammar automatically extracted from the Penn Chinese Treebank. Compared with hand-crafted grammars, treebank grammars comprise a very large number of grammar rules, the majority of which occur very infrequently. Table 5.6 shows some statistics for the functionally (but not parent-) annotated CFG rules collected from the CTB5.1 training set. At more than 8,800 rules, the automatically extracted grammar is rather large, but more than half of them — 4,782 or 53.89% — occur only once in the treebank.

Count	#Rules	%Rules
1	4782	53.89
2	1098	12.37
3	548	6.18
4	359	4.05
5	227	2.56
6-10	558	6.29
11-20	419	4.72
21-50	376	4.24
51-100	170	1.92
>100	336	3.79
all	8,873	100

Table 5.6: Statistics for the rules extracted from the training set of CTB5.1

As there is no constraint on string position in generation, in theory a grammar could allow any pair of edges to combine in the chart generator, which results in an exponential time complexity in the worst case (even though chart parsing is polynomial). If the generator is based on a wide-coverage treebank grammar, the situation is likely to become even worse, because more than one treebank grammar rule can be applied to combine the same two edges in many cases. For example, the common word 要/*need* has two POS tags AD and VV in the CTB5.1 training set, but the two POS tags lead to ADJP, ADVP, DVP, VP, VRD, PP, IP, CP, NP, INTJ and FRAG constituents, 11 edges in total by only unary rules extracted from the treebank. Some of the rules, such as $ADJP \rightarrow AD$, $NP \rightarrow VV$ are very uncommon and possibly just labelling errors. In order to cut down on the computational complexity

Complete Sentence	Coverage	ExMatch	BLEU	SSA	Time
PCFG-Full	98.69%	23.14%	0.7050	0.6644	100:49:41
PCFG-Reduce	96.66%	23.04%	0.7084	0.6663	9:21:19
Significance			$\sim$	$\sim$	
PF-PCFG-Full	97.06%	24.03%	0.7142	0.6708	6:10:31
PF-PCFG-Reduce	93.96%	23.87%	0.7165	0.6748	1:09:55
Significance			$\sim$	$\sim$	
PC-PCFG-Full	97.55%	24.67%	0.7206	0.6840	17:10:10
PC-PCFG-Reduce	94.37%	24.72%	0.7213	0.6857	5:41:00
Significance			$\sim$	$\sim$	

Table 5.7: Comparison between the reduced- and full-size treebank grammar on development data

while maintaining accuracy, it could be a strategy to filter out infrequent (potentially incorrect) rules and reduce the total number of rules in the treebank grammar.

Charniak (1996) reported that using a reduced treebank grammar which excludes the 1-count rules extracted from the Penn English Treebank, had almost no impact on the parsing results testing on WSJ text, compared with the parsing results using the full grammar. Inspired by Charniak (1996), I also run experiments using the subset of grammar rules that occur more than once in the training set. This reduces the number of rules by half. Table 5.7 compares the results for the development data generated by the three generative PCFG models trained on the full-sized and the reduced grammars. The results for completely generated sentences show that generation coverage drops by about 3 percentage points due to the grammar reduction, however the generation accuracy is almost unchanged. The results accord with the observation on PCFG parsing for English (Charniak, 1996). The last column of Table 5.7 gives the time cost (in the form of hh:mm:ss) for generating all sentences by each model running on a server with an Intel Xeon 1.86GHz CPU and 4GB memory. The time cost clearly indicates that the drop in number of grammar rules dramatically speeds up the generation process. I believe that the tradeoff in terms of coverage is worth the increase in speed.

Complete Sentence	Coverage	ExMatch	BLEU	SSA
no smooth	28.53%	**29.84%**	**0.7243**	**0.7344**
unknown smooth	82.98%	19.04%	0.7006	0.6527
all smooth	**96.24%**	17.77%	0.6945	0.6403
All Sentence	**Coverage**	**ExMatch**	**BLEU**	**SSA**
no smooth	100%	8.51%	0.4999	0.5500
unknown smooth	100%	15.80%	0.6688	0.6249
all smooth	100%	**17.10%**	**0.6867**	**0.6343**

Table 5.8: Results for various lexical smoothings by the basic PCFG model

5.4.4 Results on the Test Data

Finally, I carry out experiments on the test data using the PCFG, PF-PCFG, PC-PCFG generation models trained on the reduced treebank grammar. Table 5.8 gives results for the experiments varying lexical smoothing in the basic PCFG model. *no smooth* means that lexical rules are only applied to lexical features seen during training and no lexical smoothing is performed, which results in fairly low generation coverage because many strings produced are only partial (with unknown lexical features unrealised); *unknown smooth* means that lexical smoothing is carried out only for unknown lexical features never seen in the training set, which boosts generation coverage from 28.53% to 82.98%; and *all smooth* means that lexical smoothing is conducted for all lexical features of the input f-structure, which further increases generation coverage to 96.24%. Comparing results for the set of completely generated sentences, the *no smooth* experiment gives higher accuracy scores than the other two experiments with lexical smoothing, at the price of dramatically reduced coverage. One reason for this is that lexical smoothing is effectively a backoff for lexical productions unseen in the training data, which potentially can make wrong predictions for tagging lexical features and produce an incorrect derivation tree. But what is more important and needs to be noticed is that the set of sentences being evaluated is different in the comparison, since the PCFG generation model without lexical smoothing can only produce about 28% complete sentences for all input f-structures, which is much less than the number of complete sentences generated by the two models with lexical smoothing. And the small set of completely generated

Complete Sentence	**Coverage**	**ExMatch**	**BLEU**	**SSA**
PCFG	96.24%	17.77%	0.6945	0.6403
PF-PCFG	94.79%	18.28%	0.6989	0.6431
			∼PCFG	∼PCFG
PC-PCFG	94.63%	18.96%	0.7041	0.6508
			≫PCFG	>PCFG
All Sentence	**Coverage**	**ExMatch**	**BLEU**	**SSA**
PCFG	100%	17.10%	0.6867	0.6343
PF-PCFG	100%	17.33%	0.6905	0.6358
PC-PCFG	100%	**17.94%**	**0.6937**	**0.6445**

Table 5.9: Results for various PCFG models with *all smooth* lexical smoothing

sentences includes relatively shorter sentences with an average length of 17.8 words, contrasted with 21.8 and 22.3 words average length for complete sentences generated by the *unknown smooth* model and the *all smooth* model. The difference in sentence length also accounts for the big gap of the SSA scores between the models with lexical smoothing (0.6403 in *all smooth* and 0.6527 in *unknown smooth*) and the *no smooth* model (0.7344). Nevertheless, the model without lexical smoothing produces unsatisfactory overall results when evaluating sentences for all input f-structures. Carrying out smoothing for unknown lexical features greatly improves the overall results, and in turn, smoothing for all lexical features shows the best overall performance.

Table 5.9 gives results for the three PCFG generation models with smoothing for all lexical features. I find the same trend on the performance of the three PCFG-based generation models throughout all experiments. That is, increasing the conditioning context by the parent annotation transformation when predicting grammar rule expansions in the tree derivation improves on the accuracy of the simple PCFG model, at a cost of slightly lower generation coverage. Among the three models, the PC-PCFG model performs the best, achieving the highest BLEU score of 0.6937, SSA score of 0.6445 and 17.94% sentences exactly matching the references. The PF-PCFG model also outperforms the simple PCFG model, but the improvements in BLEU and SSA scores are not significant. This is might be because of the effect of the reduced grammar and also the noise caused by lexical

Complete Sentence	Coverage	ExMatch	BLEU	SSA
PCFG	100%	20.55%	0.7155	0.6580
PF-PCFG	100%	20.78%	0.7236	0.6627
			≫PCFG	>PCFG
PC-PCFG	100%	**21.70%**	**0.7330**	**0.6752**
			≫PF-PCFG	≫PF-PCFG

Table 5.10: Upper bound results on test data

smoothing. In order to prove this conjecture, I perform a close test experiment by obtaining the generation grammar from all of the f-structure annotated CTB5.1 trees (including development and test sets), and generating the test data based on this full-coverage grammar.

Table 5.10 gives the upper bound results, which reemphasises the earlier finding that increasing structural sensitivity significantly improves the performance of the PCFG-based generation model. Comparing the results of the PC-PCFG model with the PF-PCFG model, the PC-PCFG model performs better in all the experiments, this is most likely because the grammar rules underpinning the PCFG generation models already contain f-structure annotation equations, in consequence, augmenting the functionally annotated CFG rules with parent grammatical functions manifests a weaker discrimination than parent annotation with constituent categories. The enrichment of the additional f-structure equations in the generation rules also partially accounts for the fact that the parent annotation transformation in my PC-PCFG and PF-PCFG models does not show enhancement as prominent as in PCFG-based parsing, for example, Johnson (1998) reported the parent transformation achieved a remarkable increase of about 8% in precision and recall for parsing the English Penn-II data.

5.5 Summary

In this chapter, I have presented an approach to wide-coverage, probabilistic Chinese sentence realisation from LFG f-structures based on an automatically obtained wide-coverage f-structure annotated treebank grammar. I compared generative PCFG

models with conditional generation models. I found that proper generative PCFG models implemented in a chart-style generator overcome the low generation coverage caused by conditioning CFG productions directly on input f-structure features as used in the original proposal of Cahill and van Genabith (2006) and Hogan et al. (2007). At the same time, the generative PCFG models are comparable or even superior to the conditional generation models in terms of generation accuracy. I also investigated various ways to break down inappropriate independence assumptions imposed by the simple PCFG model. Including more structural context either on the functional or constituent level (as provided by the technique of parent annotation transformation originally proposed for PCFG parsing), is sufficient to improve the performance of the simple PCFG generation model. For the task of LFG-based generation for Chinese data, including the constituent context information proved to be more effective than including functional context in the underlying PCFG.

However, all PCFG-based generation models presented in this chapter retain the basic PCFG model's lack of lexical sensitivity. From the experience of parsing, including lexical dependencies produces major advances in PCFG-based parsers (Bikel and Chiang, 2000). It is an interesting and promising direction to migrate such techniques from PCFG parsing to generation, and further improve the PCFG models through lexicalised rules.

Another important issue is the strategy for selecting rules from the treebank grammar. At the current stage, I reduce the size of the grammar simply by rule frequency thresholding. Though this is an effective method to speed up the generation process, it is still a problem how to find the appropriate threshold of the frequency value by which to select rules. Moreover, this simple method eliminates some infrequent but still necessary rules and causes a slight drop in generation coverage. Dickinson and Meurers (2003) presented a method to detect annotation inconsistencies in treebanks and they found that removing inconsistent rules led to an improvement in parsing accuracy. It is promising to investigate the method of Dickinson and Meurers (2003) or other more sophisticated techniques for detecting and removing certain grammar rules (rather than just infrequent rules) in the future.

Chapter 6

Dependency N-Gram-Based Generation

6.1 Introduction

The PCFG-based LFG generation models described in the previous chapter can be viewed as the inverse of the parsing process, using bi-directional CFG rules. Nevertheless, the ultimate task for generation is to choose the most likely string and not the most likely syntactic tree, as in parsing. In the PCFG-based models, the generation process is driven by an input f-structure (or dependency) representation, but proceeds through the "detour" of using functionally annotated CFG (or PCFG) rules and a chart-based generator. By contrast, this section presents simple and novel Dependency-based N-gram (DN-gram)[1] probabilistic generation models, which cut out the middle-man — the CFG component — from previous approaches. Part of the research presented in this chapter has been published in Guo et al. (2008a).

[1] I use the term "DN-gram" to refer to dependency-based n-grams to distinguish them from the conventional surface-string-oriented n-gram language model.

6.2 Premises for Dependency-Based Generation

The method presented in this chapter is completely different from the Nitrogen/ HALogen (Langkilde and Knight, 1998a; Langkilde, 2000) and FERGUS (Bangalore and Rambow, 2000) approaches, which first produce a large set of potential realisation candidates using grammar rules and then rank all realisations by n-gram language models over surface strings. By contrast, my method bears a broad resemblance to the approaches of Uchimoto et al. (2000) and Filippova and Strube (2007). However, unlike their more limited approaches which concentrate on a limited set of linguistic phenomena, my approach provides a full-scale, general purpose and wide-coverage generator. The generator employs DN-gram models that estimate the likelihood of the appropriate word order *directly* from a set of bilexical labelled dependencies (or f-structures) rather than as a by-product of constructing syntactic trees. As mentioned in Section 4.2.2, grammatical relations bear an important influence on Chinese word order. For example, Chinese sentence structure is essentially SVO, modifiers tend to precede the head/predicate in noun phrases, and so forth. Based on this observation, we determine the linear order of constituents and words from input f-structures by means of n-gram models over grammatical functions or dependency labels (rather than surface word forms, as in previous language model based approaches).

It has been observed that n-gram models can be expensive to apply in generation: in order to select the most likely realisation according to an n-gram model, all alternative realisations have to be generated and the probability of each realisation according to the model has to be calculated. This can be very time-consuming if the number of alternatives turns out to be vast. In Nitrogen (Langkilde, 2000), the n-gram language model deals with trillions of alternatives. To reduce the number of possible realisations, I break down the entire generation space as defined by the input f-structure into each level of embedding within the input f-structure, i.e. the generator selects the most likely substring for each local sub-f-structure, and concatenates the substrings into a complete sentence by recursively traversing each level of embedding until it reaches the outermost f-structure. This reduction in

(1) 钱 父母 不 让 我 用 来 享乐
money parents not let me use to please
'Money, my parents do not let me use for pleasure.'

$$
f_1\left[\begin{array}{ll}
\text{PRED} & \text{'让}\langle\text{SUBJ, OBJ, XCOMP}\rangle\text{'} \\
\text{TOPIC} & f_2\left[\begin{array}{ll}\text{PRED} & \text{'钱'}\\ \text{NTYPE} & \text{common}\end{array}\right]\boxed{1} \\
\text{SUBJ} & f_3\left[\begin{array}{ll}\text{PRED} & \text{'父母'}\\ \text{NTYPE} & \text{common}\end{array}\right] \\
\text{OBJ} & f_4\left[\begin{array}{ll}\text{PRED} & \text{'pro'}\\ \text{PERS} & 1\\ \text{NUM} & \text{sg}\end{array}\right]\boxed{2} \\
\text{ADJUNCT} & \left\{f_5\left[\begin{array}{ll}\text{PRED} & \text{'不'}\\ \text{ADJ-TYPE} & \text{neg}\end{array}\right]\right\} \\
\text{XCOMP} & f_6\left[\begin{array}{ll}
\text{PRED} & \text{'用}\langle\text{SUBJ, OBJ, XCOMP}\rangle\text{'} \\
\text{SUBJ} & \boxed{2} \\
\text{OBJ} & \boxed{1} \\
\text{XCOMP} & f_7\left[\begin{array}{ll}\text{PRED} & \text{'享乐}\langle\text{SUBJ}\rangle\text{'}\\ \text{SUBJ} & \boxed{2}\\ \text{MSP} & \text{'来'}\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figure 6.1: Reentrancies representing NLDs in LFG

complexity, however, does not permit non-projective dependency- or f-structures, where the projection of a local head/predicate can be discontinuous. Nevertheless, the projectivity assumption is feasible in the task of dependency-based generation, because: (i) dependency structures of a vast majority of sentences in the languages of the world are projective (Mel'čuk, 1988) and (ii) non-projective dependencies in a language such as Chinese with relatively rigid word order, are mainly used to account for non-local dependency phenomena, which are represented as reentrancies in f-structures as exemplified in Figure 6.1. Though the reentrancies in the f-structure analysis appropriately reveal that the discourse function TOPIC in f_1 should be interpreted as the object of the embedded predicate of f_6, and the object controller in f_1 is also an argument function in the subordinate f-structures f_6 and f_7, only the antecedent functions (TOPIC and OBJ controller) are overt in the surface realisation. This means that the reentrancies for the trace functions are only necessary for

(2) 费德勒 轻松 晋级 下 一 轮 比赛
Federer easily advance next one round competition
'Federer eased into the next round.'

$$
f_1\left[\begin{array}{ll}
\text{PRED} & \text{'晋级'} \\
\text{SUBJ} & f_2\left[\begin{array}{ll}\text{PRED} & \text{'费德勒'}\\ \text{NTYPE} & \text{proper}\\ \text{PERS} & 3\end{array}\right] \\
\text{OBJ} & f_3\left[\begin{array}{ll}
\text{PRED} & \text{'比赛'} \\
\text{NTYPE} & \text{common} \\
\text{PERS} & 3 \\
\text{DET} & f_4\left[\text{PRED} \quad \text{'下'}\right] \\
\text{QUANT} & f_5\left[\begin{array}{lll}\text{PRED} & & \text{'轮'}\\ \text{NUMBER} & f_6\left[\begin{array}{ll}\text{PRED} & \text{'一'}\\ \text{NUMBER-TYPE} & \text{cardinal}\end{array}\right] & \end{array}\right]
\end{array}\right] \\
\text{ADJUNCT} & \left\{f_7\left[\text{PRED} \quad \text{'轻松'}\right]\right\}
\end{array}\right]
$$

string	费德勒	轻松	晋级	下	一	轮	比赛
position	1	2	3	4	5	6	7
f_1	SUBJ	ADJUNCT	PRED				OBJ
f_3				DET		QUANT	PRED
f_5					NUMBER	PRED	

Figure 6.2: Linearisation of grammatical functions / labelled dependencies

semantic interpretation and therefore can be removed from the input f-structure for generation, thus non-projective dependencies are transformed into simple projective dependencies.

The advantage of the projectivity assumption is that it can greatly reduce the time complexity of n-gram models. To generate a sentence consisting of n words, the search space of all possible surface string permutations is $n!$. By contrast, on the assumption of projectivity, for an f-structure with n sub-f-structures where at each local sub-f-structure f_i there are k_i local grammatical functions, the DN-gram model complexity is proportional to $k!$, where $k = \max_{i=1}^{n} k_i$. In practice, $k \ll n$ or $k < n$. To give an example, for sentence (2) in Figure 6.2 that has 7 words corresponding to 7 sub-f-structures, a naive word-based n-gram model searches among $7! = 5040$ permutations. By contrast, given the projectivity assumption, the DN-gram model

searches each local f-structure f_i, resulting in only $4!+1!+3!+1!+2!+1!+1! = 36$ possibilities in total.

6.3 DN-Gram Models

At a particular level of f-structure, the generator realises the (partial) sentence covered by that f-structure by linearising the set of GFs present at the local f-structure. For example, the set of GFs $\{\text{PRED}, \text{DET}, \text{QUANT}\}$ of f_3 in Figure 6.2 generates the string *下一轮比赛/next round of competition* in accordance with the GF sequence $\langle \text{DET}, \text{QUANT}, \text{PRED} \rangle$. In order to capture the appropriate linear order of GFs, I develop a number of DN-gram models with increasing complexity.

6.3.1 The Basic DN-Gram Model

I start with a basic DN-gram model. Different from traditional word-based language models, the DN-gram model is based on the names of GFs (including PRED) or dependency labels instead of words. Given a (sub-)f-structure containing m GFs, the DN-gram generation model searches for the best surface string S_1^m=$s_1...s_m$ generated by the GF linearisation $GF_1^m = GF_1...GF_m$, which maximises the probability $P(GF_1^m)$. Applying the chain rule, the probability of the entire GF sequence $P(GF_1^m)$ can be decomposed as:

$$\begin{aligned} P(GF_1^m) &= P(GF_1)P(GF_2|GF_1)P(GF_3|GF_1^2)\dots P(GF_k|GF_1^{k-1}) \\ &= \prod_{k=1}^{m} P(GF_k|GF_1^{k-1}) \end{aligned} \tag{6.1}$$

Under the Markov assumption, we can approximate the conditional probability of the individual GF_k by n-grams: $P(GF_k|GF_1^{k-1}) \approx P(GF_k|GF_{k-n+1}^{k-1})$, thus the probability of the complete GF sequence $P(GF_1^m)$ is computed according to Eq. (6.2).

$$P(GF_1^m) = P(GF_1...GF_m) = \prod_{k=1}^{m} P(GF_k|GF_{k-n+1}^{k-1}) \tag{6.2}$$

To estimate the DN-gram probabilities, I build a dependency/GF corpus as

follows:

1. I automatically build the f-structure bank from the CTB as described in Section 2.4.

2. I linearise the unordered set of GFs at each level of embedding in each f-structure by associating the surface string position (numerical word offset from start of the sentence) with the local predicate.

3. GF sequences are collected from the f-structures as in Figure 6.2.

From this corpus of GF sequences, I obtain simple relative frequency estimates for the parameters of the DN-gram model by taking counts:

$$P(GF_k|GF_{k-n+1}^{k-1}) = \frac{Count(GF_{k-n+1}^{k-1}GF_k)}{Count(GF_{k-n+1}^{k-1})} \tag{6.3}$$

6.3.2 Factored DN-Gram Models

The most basic DN-gram model over bare GFs assumes that generation at each sub-f-structure is independent of any other (sub-)f-structure (in the larger f-structure). To weaken this independence assumption, I integrate limited contextual and fine-grained lexical information into several factored models. One way to factor the DN-gram model is to take into account contextual features associated with the f-structure f_i which yields the string in question. Eq. (6.4) additionally conditions the distribution on the parent GF label $Parent_i$ of the current f-structure f_i, and Eq. (6.5) conditions the distribution on the head word (or predicate) $Head_i$ of f_i.

$$P^p(GF_1^m) = \prod_{k=1}^{m} P(GF_k|GF_{k-n+1}^{k-1}, Parent_i) \tag{6.4}$$

$$P^h(GF_1^m) = \prod_{k=1}^{m} P(GF_k|GF_{k-n+1}^{k-1}, Head_i) \tag{6.5}$$

Another type of factored model includes additional features of the GFs being ordered. I do this by augmenting the label of GF_k with atomic-valued features[2] from

[2]I use the term "feature" for the pair of attribute and its corresponding value in f-structure.

the local sub-f-structure that is assigned to the attribute GF_k. Eq. (6.6) stands for this featured model, where $Feat_k$ represents a variety of atomic-valued features specifying the local f-structure, such as TENSE, MOOD, NUM etc. PRED is a special attribute in f-structure whose value is a semantic form.[3] In contrast with other atomic features having a closed set of values, semantic forms are associated with an open number of lexical entries. I separate PRED from other atomic-valued features and develop a truly lexicalised DN-gram model (Eq. 6.7).

$$P^f(GF_1^m) = \prod_{k=1}^{m} P(Feat_k|Feat_{k-n+1}^{k-1}) \tag{6.6}$$

$$P^l(GF_1^m) = \prod_{k=1}^{m} P(Lex_k|Lex_{k-n+1}^{k-1}) \tag{6.7}$$

Table 6.1 exemplifies various DN-gram models for the local f-structure f_3 in Figure 6.2.

	DN-grams	Condition
basic (P)	DET $\prec$ QUANT $\prec$ PRED	
parent (P^p)	DET $\prec$ QUANT $\prec$ PRED	OBJ
head (P^h)	DET $\prec$ QUANT $\prec$ PRED	'比赛'
feat (P^f)	DET[ ] $\prec$ QUANT[ ] $\prec$ PRED[NTYPE=common,PERS=3]	
lex (P^l)	DET[PRED='下'] $\prec$ QUANT[PRED='轮'] $\prec$ PRED[PRED='比赛']	

Table 6.1: Examples of DN-grams for f_3 in Figure 6.2

In addition, the factored models can be combined in different ways. As lexicalised models are likely to suffer from severe data sparseness, I create combined models by linearly interpolating various factored DN-gram models P^{Fi} incrementally (rather than by using chain rules) as in Eq. (6.8):

$$P^{FC}(GF_1^m) = \sum \lambda_i P^{Fi}(GF_1^m) \tag{6.8}$$
$$where \quad \sum \lambda_i = 1$$

Finally, to overcome the problem of sparse data, all the individual and combined factored DN-gram models P^F are linearly interpolated with the basic DN-gram

[3] In the implementation, the value of PRED is only the predicate name (or lemma) without specifying its argument-list.

model P for smoothing (Eq. 6.9).

$$\hat{P}^F(GF_1^m) = \lambda P^F(GF_1^m) + (1-\lambda)P(GF_1^m) \tag{6.9}$$

6.4 DN-Gram-Based Generation Algorithm

Assuming projective dependencies, the linearisation of GFs at the given f-structure is reduced to the linearisation at each local sub-f-structure. Specifically, the algorithm generates partial strings covered by subsidiary f-structures according to the DN-gram probabilities and combines all the partial strings into a longer (partial) string yielded by the next outer f-structure. For example, in Figure 6.2 the partial strings covered by f_2(*费德勒/Federer*), f_3(*下一轮比赛/next round of competition*), f_7(*轻松/easily*) are generated first, and f_1's local predicate is realised as the word (*晋级/advance*). Then all the partial strings are arranged in correct order by the DN-gram model to achieve the final sentence realisation for f_1 (*费德勒轻松晋级下一轮比赛*).

In summary, given an input f-structure f, the core algorithm of the generator recursively traverses f in a bottom-up fashion, and at each level of embedding with f-structure f_i:

1. instantiates the local predicate PRED_i of f_i
2. calculates the linearisations of the set of GFs present at f_i by DN-gram models
3. finds the most probable GF sequence among all possible linear orders by Viterbi search
4. generates the surface string s_i yielded by f_i according to the best GF sequence

Finally, function words are inserted at particular positions by heuristics at each sub-f-structure. If an atomic-valued feature representing a function word (as listed in Table 5.2) is present in a sub-f-structure f_i, the algorithm inserts the corresponding function word at one of four possible positions: (i) before the local predicate PRED_i; (ii) after the local PRED_i; (iii) before the entire string s_i yielded by f_i; (iv) after

the string s_i. For example, auxiliary particles of the MSP attribute precede the local PRED, as 而 in sentence (3) generated immediately before f_1's predicate word 去. The structural particle 的 follows the modifier of a noun phrase, as in sentence (4) 的 is inserted after the string corresponding to adjunct f_2 has been generated.

(3) 随 之而去
follow it *to* go
'go with it'

$$f_1\begin{bmatrix}\text{PRED} & \text{'去'}\\ \text{MSP} & \text{'而'}\\ \text{ADJUNCT} & \left\{ f_2\begin{bmatrix}\text{PRED} & \text{'随'}\\ \text{OBJ} & f_3\begin{bmatrix}\text{PRED} & \text{'之'}\\ \text{PERS} & 3\\ \text{PRON-TYPE} & \text{person}\end{bmatrix}\end{bmatrix}\right\}\end{bmatrix}$$

(4) 不 小 的 差距
not small *DE* distance
'a not small distance'

$$f_1\begin{bmatrix}\text{PRED} & \text{'差距'}\\ \text{PERS} & 3\\ \text{NTYPE} & \text{common}\\ \text{ADJ-REL} & \left\{ f_2\begin{bmatrix}\text{PRED} & \text{'小'}\\ \text{de} & +\\ \text{ADJUNCT} & \left\{ f_3\begin{bmatrix}\text{PRED} & \text{'不'}\\ \text{ADJ-TYPE} & \text{negative}\end{bmatrix}\right\}\end{bmatrix}\right\}\end{bmatrix}$$

6.5 Experiments and Results

Experiments are carried out on the same test/development/training splits as the PCFG-based generation experiments (see Section 5.4.1 for details) but this time including all sentences without length limitation. Table 6.2 shows some statistics

	Test	Develop	Train
total num of sentences	1,718	1,592	15,094
max sentence length (#words)	117	159	240
avg sentence length (#words)	30	29	26
total num of sub-f-strs	24,036	21,204	182,218
num of sub-f-strs per sentence	13.99	13.32	12.07
max length of sub-f-str (#gfs)	14	35	35
avg length of sub-f-str (#gfs)	2.90	2.89	2.92

Table 6.2: Properties of the experimental data

for the experimental data.

BLEU score, NIST simple string accuracy (SSA) and percentage of exactly matched sentences are adopted as evaluation metrics. Statistical significance tests are also performed for BLEU and SSA scores. As the dependency-based generation algorithm guarantees that all input f-structures can produce a complete sentence, coverage-dependent evaluation (as used in evaluating grammar-based generation in Chapter 5) is not necessary in these experiments.

6.5.1 Order of the DN-Gram Models

The DN-gram models are created using the SRILM toolkit (Stolcke, 2002) with Good-Turing discounting and Katz backoff for smoothing. Table 6.3 shows the results for the basic DN-gram models with order from 2 to 5 on the development data. In addition to the evaluation metrics BLEU, SSA scores and exact matches, I also computed the perplexity of different order DN-gram models on the development set. Perplexity is the most common intrinsic evaluation metric to measure the performance of n-gram models. The intuition behind perplexity is that given two probabilistic models, the better model is the one that has the tighter fit to the test data, or predicts the details of the test data better. Better predication can be measured by looking at the probability the model assigns to the test data: the better model assigns a higher probability to the test data. For the sentence $S = s_1 s_2 \dots s_m$ realised by the GF sequence $GF_1 GF_2 \dots GF_m$, the perplexity is the probability that

the GF n-gram model assigns to that sentence, normalised by the number of GFs:

$$PPL(GF_1^m) = P(GF_1 \dots GF_m)^{-\frac{1}{m}} = \sqrt[m]{\prod_k^m \frac{1}{P(GF_k|GF_{k-n+1}^{k-1})}} \tag{6.10}$$

Note that because of the inverse in equation (6.10), the higher the conditional probability of the GF sequence, the lower the perplexity.

	Evaluation Metrics				Significance Test	
DN-gram	ExMatch	BLEU	SSA	PPL	BLEU	SSA
2-gram	11.34%	0.5753	0.5361	3.5795		
3-gram	11.41%	0.5817	0.5417	3.0333	$>$ 2-gram	$>$ 2-gram
4-gram	**11.66%**	**0.5826**	**0.5427**	2.9034	$\sim$ 3-gram	$\sim$ 3-gram
5-gram	11.59%	0.5821	0.5424	**2.8756**	$\sim$ 4-gram	$\sim$ 4-gram

Table 6.3: Results for different order of basic DN-grams on the development set

For the basic (i.e. non-factored, non-lexicalised) model, all the scores for the 3-gram model are much better than those for the 2-gram model, and in turn the 4-gram model is slightly better than 3-gram. There is a contradiction between the intrinsic perplexity measure and other scores on the 5-gram model. Although the perplexity of the 5-gram model is the lowest, i.e. the conditional probability of the GF sequences produced by the 5-gram model is the highest, no improvement is made in any of the metrics for evaluating generation accuracy. However, despite the variance, the statistical significance tests (with significance level p=0.05) show that only the 3-gram model significantly improves on the 2-gram model in terms of BLEU and SSA scores. There is no significant difference between the performance of the 3-, 4- and 5-gram models. Similar tests were also carried out for the various factored DN-gram models on the development set. Rankings of the accuracy scores are somewhat changeable among the 3-, 4- and 5-gram models in different factored models. Nevertheless, all the models with order higher than 2 are significantly better than the bigram models, but with more features incorporated in the DN-gram models, less difference remains between the performance of the models with different order. As this is the case, I implement trigram models on the test set data throughout the experiments.

6.5.2 Evaluation of Features

Besides the lexical value of PRED, the featured DN-gram model exploits 17 atomic-valued features presented in the f-structure resources, listed in Table 6.4 (refer to Appendix A for a detailed description of each attribute and its corresponding values). These atomic-valued features make different contributions to the DN-gram generation model while linearising GFs. Table 6.4 evaluates the importance of each feature on the development set. The first line is the DN-gram model incorporating all atomic-valued features except PRED, viz. the $feat$ model represented by Eq. (6.6), which I take as a baseline model in this experiment. I discard one feature each time,[4] and compare the model incorporating the rest of the features to the baseline model. In Table 6.4, $<$ means the result is significantly worse than that of the baseline model (with $p\text{-}value$=0.05), $\sim$ means the decrease (or increase) in the score is not significant.

Atomic-valued Features	ExMatch	BLEU	SSA	BLEU	SSA
All Features	15.25%	0.6467	0.5830		
ADJ-TYPE	13.30%	0.6174	0.5588	$<$	$<$
NTYPE,NUM,PERS	13.80%	0.6361	0.5728	$<$	$<$
de	15.00%	0.6407	0.5799	$<$	$<$
MSP	15.00%	0.6436	0.5784	$<$	$<$
PRON-TYPE,NUM,PERS,GEND	15.12%	0.6450	0.5818	$<$	$\sim$
dr	15.12%	0.6460	0.5823	$\sim$	$\sim$
MOOD	15.25%	0.6466	0.5817	$\sim$	$\sim$
ASPECT	15.25%	0.6460	*0.5839*	$\sim$	$\sim$
PRECOORD-FORM	15.25%	0.6466	*0.5831*	$\sim$	$\sim$
NUMBER-TYPE	15.12%	*0.6471*	*0.5838*	$\sim$	$\sim$
di	15.25%	*0.6471*	*0.5836*	$\sim$	$\sim$
CLAUSE-TYPE	15.19%	*0.6474*	*0.5844*	$\sim$	$\sim$
vnv,VNV-FORM	*15.31%*	*0.6467*	*0.5833*	$\sim$	$\sim$

Table 6.4: Evaluation of atomic-valued features on the development set

According to the statistical significance for differences between the results of the featured DN-gram models, the 17 features are divided into two groups: (i) the features in the upper 5 lines have a crucial effect upon the DN-gram generation model, for ignoring one of them from local f-structures, the BLEU and SSA scores

[4]Some features always come along with another feature, e.g. NUM, PERS and GEND occur with NTYPE or PRON-TYPE, and VNV-FORM occurs with vnv, thus I delete these features together.

for the resulting realisations become significantly worse compared to the baseline *feat* model. Among those features, ADJ-TYPE is the most important feature for linearising GFs, since ADJ-TYPE reflects semantic types of modifiers, such as time, location, direction, and purpose etc., which to some extent decide the order of modifiers in Chinese. NTYPE and PRON-TYPE indicate that the part-of-speech of the predicate is a noun or a pronoun, which is also a factor in GF linearisation. For example, a general principle of the sequence of attributive modifiers in Chinese is: pronoun $\prec$ determiner $\prec$ quantifier $\prec$ adjective $\prec$ noun (Zhu, 1982, pp.151), as in sentence (5).

(5) 他 那 件 新 皮子 大衣
his that CLS new leather coat
'His new leather coat.'

The feature DE, as mentioned before, is a descriptive indicator attached to attributive modifiers and the DE-phrase usually precedes other attributive modifiers without DE (Zhu, 1982, pp.151), e.g.

(6) 上个月 *的* 一次同学 聚会
last month *DE* one classmates gathering
'A classmates gathering last month'

MSP is another feature indicating function words prefixed to the verb phrase. A verbal predicate bearing an MSP feature usually follows another verbal or prepositional phrase, e.g.

(7) 随着 年龄*而* 改变
along with age *MSP* change
'change with age'

(ii) the rest of the features under the dividing line in Table 6.4 do not show a great influence in determining the linear order of the GFs, because after deleting one of those features, the performance of the generator does not show a significant deterioration. On the contrary, omitting some of the features has the effect of increasing the BLEU or SSA score (though not significantly). Two reasons may account for why these features do not have a significant effect on generation accuracy. First, some features do not affect the position of the GF in the sentence, such as the

feature VNV is only used for verb compounds of the form A-not-A or A-one-A, e.g. 能不能/*can not can* used in interrogative sentences, 比一比/*compare one compare* reflecting frequency or duration of the action. Another reason is that some features are not common phenomena, such as the PRECOORD-FORM encoding list markers or the first conjunction in correlative conjunctions, e.g. 不但/*not only* in the phrase 不但/*not only* ... 而且/*but also* ..., which occurs only 258 times in the training f-structures, compared with 43,351 times for the frequent feature ADJ-TYPE.

I choose the features that significantly impact on the generation performance as essential features, viz. the features in the upper 5 lines in Table 6.4. Table 6.5 shows the generation results of the DN-gram model which augments the bare GF with the essential features, on the development and test set, respectively. Both the development and test data show that DN-gram models incorporating only essential features performed nearly the same as the model incorporating all atomic-valued features. Therefore, I use only essential features in the featured DN-gram generation model in the following experiments on the test data.

	Development Set			Test Set		
	ExMatch	BLEU	SSA	ExMatch	BLEU	SSA
All Features	15.25%	0.6467	0.5830	12.28%	0.6526	0.5711
Essential Features	15.37%	0.6480	0.5860	12.17%	0.6522	0.5678

Table 6.5: Results for the DN-gram model with essential features

6.5.3 Results on the Test Data

Table 6.6 lists the generation results of different DN-gram models on the test data (for all sentence lengths), where $\gg$ indicates significance at level p=0.005, $>$ indicates p=0.05 and $\sim$ means no significant difference. The value of interpolation weights λ for the individual and combined factored ND-gram models are set by testing on the development data and listed beneath the models.

The experiments are conducted in a series of cascades. I first generate the sentences from input f-structures of the test data by the basic and each individual factored ND-gram model. It is not a surprise that the fully lexicalised model *lex*

	DN-gram Model	ExMatch	BLEU	SSA
1.	basic	8.91%	0.5881	0.5268
2.	parent	9.72%	0.6137	0.5384
	(λ=0.8)		$\gg 1$	$\gg 1$
3.	head	10.59%	0.6320	0.5471
	(λ=0.8)		$\gg 2$	$\gg 2$
4.	feat	12.17%	0.6522	0.5678
	(λ=0.9)		$\gg 3$	$\gg 3$
5.	lex	14.38%	0.6861	0.6120
	(λ=0.9)		$\gg 4$	$\gg 4$
6.	lex+parent	15.42%	0.7010	0.6161
	(λ_1=0.8,λ_2=0.15)		$\gg 5$	~ 5
7.	lex+feat	16.24%	0.7041	0.6219
	(λ_1=0.8,λ_2=0.15)		~ 6	> 5
8.	lex+head	16.82%	0.7121	0.6248
	(λ_1=0.7,λ_2=0.2)		$\gg 7$	$\gg 5$
9.	lex+head+parent	16.01%	0.7063	0.6220
	(λ_1=0.6,λ_2=0.15,λ_3=0.15)		$\ll 8$	~ 8
10.	lex+head+feat	**17.17%**	**0.7177**	**0.6300**
	(λ_1=0.6,λ_2=0.15,λ_3=0.15)		> 8	> 8

Table 6.6: Results for different DN-gram models on the test set

greatly surpasses other individual models in all evaluation metrics. Thus I select the lexicalised model as a new baseline model, and combine it with the model augmented with atomic-valued features $feat$, the two conditional models *parent* and *head*, resulting in models 7, 6 and 8, respectively. In turn, the model *lex+head* combining the conditional head word model with the lexicalised model outperforms the other two combined models and hence is chosen as the new baseline model. Again, the *lex+head* model is interpolated with the remaining two factors *parent* and $feat$ into the more complex models 9 and 10. However, this time the results show that the additional conditioning feature of parent GF does not improve the generation performance. This can be interpreted as the parent GF being too general to predict the fine distinctions between the f-structures with different predicates that govern the linear order of GFs in the local f-structure. On the contrary, it counteracts the discrimination effect of the head word. Model *lex+head+*$feat$ which incorporates more features specifying the local f-structure further improves the results and achieves the best 0.7177 BLEU score and 0.6300 SSA score. The best results

are achieved on the basis of two lexicalised models *lex* and *head*, which indicates that the unique word form plays the most important role in the DN-gram based generation for Chinese.

Example realisations generated by different DN-gram models for a reference sentence (8) are listed below.

(8) 每到 假日 ，圣克里斯多福 教堂 前 的 人行道 上 就 有
Every holiday , St.Christopher Cathedral before DE pavement on AUX have
人 来此 摆设 英文 报摊 。
somebody come place English bookstall .
'There will be English newspaper stalls on the pavement before the St.Christopher Cathedral every holiday.'

lex+head: 每到假日，圣克里斯多福教堂前的人行道上就有人来此摆设英文报摊。

lex+feat: 前圣克里斯多福教堂的人行道上每到假日，就有人来此摆设英文报摊。

lex+parent: 每到假日，上前圣克里斯多福教堂的人行道就有人来此摆设英文报摊。

lex: 每到假日，就上前圣克里斯多福教堂的人行道有人来此摆设英文报摊。

feat: 前圣克里斯多福教堂的人行道上，每到假日就有人摆设英文报摊来此。

head: 圣克里斯多福教堂前的人行道上，每到假日就有人摆设英文报摊此来。

parent: 上前圣克里斯多福教堂的人行道，每到假日就有人摆设英文报摊来此。

basic: 上前圣克里斯多福教堂的人行道每到假日就有，人摆设英文报摊来此。

The above sentence realisations show that lexicalised models do have a great effect on the performance of the generator. For example, the *head* model conditioned on head words captures the correct order of locative phrases. In Chinese, a locative phrase is usually predicated by a locative marker following a noun phrase to indicate location, position, time or quantity, as in phrases (9) and (10):

(9) 圣克里斯多福 教堂 前
St.Christopher Cathedral *before*
'before the St.Christopher Cathedral'

(10) 人行道 上
pavement *on*
'on the pavement'

The order of locative phrases is opposite to that of normal prepositional phrases, in

which the preposition is the first constituent. However as locative and prepositional phrases share the same predicate-argument structure PRED⟨OBJ⟩, the correct order of locative phrases is consequently overwhelmed by larger numbers of prepositional phrases in the basic DN-gram model. By contrast, the *head* model is capable of distinguishing locative phrases from prepositional phrases by the predicate lexeme.

Likewise, the fully lexicalised model *lex* takes the lexeme of the GF into account while determining the order of GFs, and outperforms other DN-gram models. For instance, only the individual lexicalised model and models combined with the *lex* model establish the correct order of the phrase (11) in the above examples.

(11) 有 人 来此 摆设 英文 报摊
have somebody *come here* set up English bookstall
'Somebody will come to set up a bookstall of English newspapers.'

6.6 Summary

6.6.1 Comparison between PCFG and DN-Gram Models

I described two theoretically distinct approaches to Chinese sentence realisation from LFG f-structures. In the previous chapter, I present a more conventional way of generating sentences through application of functionally-annotated grammar rules to construct the most probable syntax or derivation trees, which is generally viewed as the inverse process of parsing. This chapter solves the generation problem by a more direct approach of mapping from the input semantic relations to surface strings by dependency-based n-gram models. As the two approaches reach the same goal by different routes, the question is which is the better route? Though the answer to the question may vary in the light of different applications and languages, a general comparison between the PCFG- and DN-gram-based generation models can be made in terms of the following aspects.

Simplicity: The DN-gram model linearises dependencies from input representations directly, obviating complex syntactic tree structures. In this sense, the dependency-based method provides simplicity and reduces overhead costs in building

a large number of grammar rules (either hand-crafted or automatically acquired from treebanks), which is inevitable in the annotated CFG grammar-based approach.

Coverage: The quality of grammar rules is a crucial factor in PCFG-based generation models. Compared to hand-crafted grammars, treebank grammars have the advantage of being large-scale and reusable. However, due to the size of available treebanks, there is a possibility that some grammar rules are never seen in the training set, which causes the grammar-based generator to fail to generate a complete sentence. This is much less of a problem in the dependency-based generator, as the linearisation operates over a small number of general GF types (rather than a large number of possibly highly specialised annotated CFG rule types). In this regard, the DN-gram models are more robust than the PCFG models, even those based on treebank grammars.

Accuracy: Table 6.7 gives the generation accuracy of the PCFG and DN-gram generation models evaluating on the test sentences up to 40 words in length. The results of PCFG-based generation are obtained by the c-structure parent annotation PCFG (PC-PCFG) model trained on the reduced grammar. For dependency-based generation, results of two models are given: the $feat$ model only employs GF labels and general atomic features of the given f-structures; the $lex{+}head{+}feat$ model additionally incorporates the lexical information in the n-grams and achieves the best results among all the various DN-gram generation models.

Model	Coverage	ExMatch	BLEU	SSA
PC-PCFG	94.63%	18.96%	0.7041	0.6508
feat	100%	16.26%	0.6703	0.6313
lex+head+feat	100%	22.16%	0.7358	0.6862

Table 6.7: Comparison between PCFG and DN-gram models on the test data ($\leq$40 words)

The results show that without lexical information, the PCFG model incorporating syntactic structures markedly outperforms the relatively simple DN-gram model with regard to generation accuracy. However, lexicalisation of the dependency n-

grams substantially improves the performance of the DN-gram model and finally surpasses the unlexicalised PCFG model. As there is still plenty room to improve the performance of the PCFG-based model through lexicalisation of the generation grammar, I expect that future work on lexicalisation will increase the accuracy of the PCFG-based generator to the level comparable with or even better than the DN-gram model.

Time Complexity: In theory, the worst case computational complexity of DN-gram-based models is $O(n!)$ to generate all the permutations and to find the most probable one, for linearising n GFs/dependencies in a local f-structure. The implementation, using dynamic programming techniques, solves the problem in time $O(n^2 2^n)$ (Bellman, 1962). In a local chart containing n edges, if a combination happens between any two edges, 2^n edges will be generated in the chart. From a theoretical perspective, the time complexity of the DN-gram dependency generator is comparable to that of the PCFG-based chart generator: they are both exponential. However, I find that the execution times of the two approaches are very different in practice. The DN-gram generation model generates one realisation with the highest probability for each local f-structure, and in each local f-structure the number of GFs is the main factor affecting the time cost. This number turns out to be small (around 2.9 on average for both test and development data), which ensures that the DN-gram generator runs very fast. Running on an Intel Pentium IV server with a 3.80GHz CPU and 3GB memory, it only took the DN-gram models approximately 4 minutes to generate all 1,718 sentences of the CTB5.1 test set. By contrast, generating the subset of test sentences with length no more than 40 words took the various PCFG-based models from a few hours to a few days. The main reason is that the PCFG generator tries to produce all alternative syntactic trees licensed by the generation grammar. Even with carefully hand-crafted grammars, it is possible that several grammar rules can be applied to combine the same edges, and hence more than one new edge (with different categories) is generated from the same source sub-f-structures. The case becomes worse when the generation grammar is

automatically extracted from a large-scale treebank. An extreme example is given in Section 5.4.3 where the simple word 要/*need* can generate a total of 11 edges by only lexical rules and unary rules of the CTB treebank grammar. Overgeneration excessively increases the number of distinct edges in the chart and sharply aggravates the time complexity of the PCFG-based chart generator. In order to curtail the time cost, refinement of the treebank grammar is necessary for the PCFG-based generator.

6.6.2 Conclusion and Future Directions

In the previous and in this chapter, I have described two different models for the task of sentence generation from LFG f-structures for Chinese. As no published results for general-purpose, wide-coverage, probabilistic sentence generation for Chinese are available, it is not easy to compare my generators directly to other systems. An internal comparison between the two methodologies shows that the DN-gram generation model is attractive due to the advantages of simplicity, efficiency, complete coverage and competitive accuracy. The virtue of the PCFG-based generation model is that it produces not only surface strings but also syntax trees, and therefore it might be interesting to some applications where more syntactic information is necessary. Theoretically, by incorporating complex syntactic tree structures, the PCFG-based models have the potential to generate sentences of superior quality, though refinement of grammar rules and more sophisticated techniques such as lexicalisation of PCFGs need to be explored to further improve the performance of the PCFG-based generation model.

When analysing the errors in the generation output, I found that neither of the two types of generators handles coordination structures very well. Due to lack of inflection and case markers, coordination is a common phenomenon in Chinese f-structures: a total of 7,377 coordinates (4.32 per sentence) occur in the f-structures for the development set. There are three different types of coordination structures attested in the data:

- Syntactic coordinates, but not semantic coordinates, such as:

  (i) 投资 百万 兴建 这个 工程
  invest million build this construction
  'invest million yuan to build the construction'

- Syntactic and semantic coordinates, but usually expressed in a fixed order, for instance:

  (ii) 改革 开放
  reform opening-up
  'reform and opening up'

- Syntactic and semantic coordinates, which can freely swap orders, e.g.

  (iii) 充沛的 精力 和 敏捷的 思维
  plentiful energy and quick thinking
  'energetic and agile'

In the current systems, I only keep the most probable realisation for each input f-structure. An alternative method in line with the generate-and-select paradigm, could pack all the locally equivalent edges or sequences into equivalence classes at the generation stage. Packing locally equivalent edges/sequences also minimises the number of edges/units produced for the local f-structure, and has the effect of reducing the time complexity. The selection module can simply give n-best candidate realisations for the coordination, and more than one acceptable reference can be used for evaluation. Or, all realisations can be re-ranked by a separate statistical model, such as a language model. The post-processing language model possibly also helps to reduce some errors caused in the previous generation stage, for instance, the realisation of function words in fixed phrases. As shown in (12), the function word 之 is incorrectly generated as 的. This is because both function words share the same part-of-speech (DEG) in CTB, but 的 has a much higher frequency than 之 in Chinese text and thus has a higher probability to be generated.

(12) a. 万物 之 中
all things DE in
'among all things'

b. *万物 的 中
all things DE in

At present, though the dependency-based generator is more developed, there is nonetheless room for further improvements. Results of the DN-gram models (Table 6.6) show that features integrated in the n-grams play a crucial role in determining the linear order of GFs. One disadvantage of the approach presented in this thesis is the need for manual feature selection, and so far a relatively small number of features have been used (even though they effectively improve on the basic model). Encouraged by the convincing results for realisation ranking using log-linear models, I expect the dependency-based generation models can be improved by taking advantage of the flexibility of log-linear models to combine more types of features. For instance, a useful feature might be the number of GFs in the current f-structure. This could help capture the properties of some languages in which the length of modifier has an effect on the constituent order (e.g. in the placement of modifiers or coordinates). Another useful feature might be the depth of the current sub-f-structure, which indicates whether it is a main clause or subordinate clause.

The DN-gram generation models were implemented within the formalism of LFG, however they are general-purpose models and suitable for any bi-lexical labelled dependencies or argument-and-relation-type representations, such as the labelled feature value structures used in HALogen (Langkilde, 2002) and the functional descriptions in the FUF/SURGE system. Therefore, it would be also interesting to apply the n-gram generation models to e.g. dependency representations as used in training state-of-the-art dependency parsers (Nivre, 2006).

Chapter 7

Conclusions

7.1 Thesis Summary

The thesis has presented two fundamental NLP tasks — parsing and generation for Mandarin Chinese within the LFG framework. The work is part of the GramLab multilingual grammar acquisition project to automate wide-coverage, deep, probabilistic LFG development based on existing treebanks. Original work carried out in the GramLab project on English (and to a limited extent on Chinese) underlies the research reported in the thesis. The successful migration and adaptation of the automatic f-structure annotation and parsing method from English to Chinese language and treebank data has proven the effectiveness, portability and language independence of the treebank-based deep grammar acquisition methodology. Still, substantial improvements and extensions to the original generic approaches and models have been achieved in both of the two NLP tasks on which the thesis concentrates:

LFG-Based Chinese Parsing

In order to parse Chinese sentences into proper f-structures, the main achievements described in this thesis include:

- I have carried out a thorough investigation on Chinese core linguistic phenomena and provided appropriate LFG analyses for particular Chinese construc-

tions from a computational perspective. Accordingly, I revised and enlarged (in cooperation with PARC) the Chinese dependency gold standard which now includes 250 f-structures for the CTB5.1 data.

- I have overhauled and substantially extended the preliminary f-structure annotation algorithm of Burke et al. (2004), which leads to a considerable increase in both grammar coverage and accuracy.

- I have developed a novel conversion-based f-structure acquisition algorithm by means of an intermediate unlabelled dependency structure. Taking advantage of the two-level syntactic representation of LFG, the method enforces a clear separation between predicate extraction and function labelling, which simplifies the annotation process and increases the robustness of the annotation algorithm.

- Based on a thorough investigation of NLD phenomena in Chinese as represented in the CTB, I have presented a hybrid NLD recovery algorithm that integrates a few heuristic rules and two statistical models based on subcat frames and NLD paths, both of which are learned from the automatically converted f-structure bank. NLD recovery turns incomplete, proto-f-structures produced from parser output trees into complete proper-f-structures.

LFG-Based Chinese Generation

- Inspired by the original proposal of Cahill and van Genabith (2006), I designed proper generative models with simple or augmented PCFGs implemented in a chart-style generator. The generative PCFG models outperform the original conditional model for the LFG-based generation task in that: (i) it overcomes the low coverage that the conditional probabilistic model suffers from; and (ii) it includes more contextual information in the generation grammar to weaken independence assumptions in the simple PCFG and improves on the generation accuracy.

- Considering that word order in Chinese is rather rigid, I proposed a direct generation approach to linearise GF units in the given f-structure by n-gram models rather than constructing constituent trees via application of grammar rules. The dependency-based n-gram model demonstrates superiority over the grammar-based generation model in both generation accuracy and efficiency for the Chinese generation task.

7.2 Future Work

There are still multiple interesting problems related to improving, extending and evaluating the specific models discussed in this thesis.

Parsing into F-structures

Cahill et al. (2008) made an extensive comparison between the automatically acquired wide-coverage LFG resources with two carefully hand-crafted constraint-based grammars — RASP and XLE. The comparison shows that the treebank-based, automatic LFG f-structure annotation algorithm together with state-of-the-art "shallow" syntactic parsers outperforms the "deep" hand-crafted wide-coverage grammars and parsing systems testing on English data. At the moment, only an approximate comparison can be made between my treebank-based automatically acquired LFG resources and the hand-crafted XLE LFG grammar developed at PARC using the rather small-scale Chinese gold-standard dependency bank data. Recently, there have been a number of on-going projects on Chinese unification-based grammar development, including Chinese HPSG grammars developed at the Universities of Tokyo and Saarland, respectively. It is worthwhile conducting a more comprehensive comparison of treebank-induced with hand-crafted, deep, wide-coverage grammars on Chinese data.

NLD Recovery

I have presented a post-processing method to recover NLDs at the level of f-structure for Chinese parser output trees. As introduced in Section 3.3, there are two other strategies for NLD recovery: pre-processing methods to introduce empty nodes in the input string before parsing and in-processing methods to integrate NLD recovery into a PCFG or history-based parser. The two methods are also attractive and worthwhile to be tested on Chinese data, especially because Chinese syntactic parsing is fairly hard and has not yet achieved a level comparable to e.g. parsing English. I expect that including NLD information into the parsing grammar or input strings would effectively improve the performance of Chinese parsers.

Grammar-Based Generation

The thesis has presented proper generative PCFG models for Chinese generation by applying the f-structure annotated CFG rules automatically acquired from the CTB, and demonstrated that the simple PCFG generation model can be improved by including more structural context to break down inappropriate PCFG independence assumptions. Informed by recent advances in PCFG parsing, lexicalising CFG rules are an alternative way to increase parsing accuracy over simple PCFGs. To the best of my knowledge, lexicalised parsers (Bikel and Chiang, 2000) outperform unlexicalised parsers for Chinese. I believe that there is a similar scope for improvement in PCFG-based generation: lexicalisation of generation grammar rules has the potential to further improve generation accuracy over the unlexicalised models described in this thesis.

DN-Gram-Based Generation

I have presented dependency-based n-gram generation models to directly linearise GFs so that the corresponding strings can be generated in the appropriate order. The units/grams used in the DN-gram model combine function labels, lexical items and other features from the input f-structure. Compared with n-gram models, log-linear models are more powerful in that they are capable of integrating arbitrary

global features or overlapping features without assuming independence among them. It is worthwhile to implement a log-linear model (instead of n-gram model) combining a wider variety of features from the given f-structure to more precisely predict the word order for Chinese generation.

In this thesis, the DN-gram generation models are designed and implemented within the LFG framework and applied to the task of generating Chinese. However, the underlying methodology is language-independent, general-purpose, and suitable for any labelled bi-lexical dependencies or typed predicate-argument representations. The models have been applied to Penn-II treebank data for generating English sentences from f-structures, and the results show that the method generalises well to different languages and data sets (Guo et al., 2008a). It would also be interesting to apply the n-gram generation models to more generic dependency structures (rather than LFG f-structures), e.g. dependency representations as used in training state-of-the-art dependency parsers (Nivre, 2006).

In NLG tasks, the input to a sentence realiser is an abstract representation without surface and language-specific information, such as punctuation. The DN-gram generation models presented in this thesis linearise all dependency relations given in the input f-structure, and thus are incapable of generating punctuation marks if they are not present in the input representation. However, punctuation is one of the most important structural elements in written language and vital to the meaning of a sentence. For the purpose of punctuation generation, a separate (hidden-ngram or log-linear) model can e.g. be designed as a post-processor to insert punctuation marks into the sentences generated by the DN-gram models.

Other NLP Applications

The thesis focuses on two basic NLP tasks: (LFG-based) parsing and generation. The linguistically rich two-level syntactic representations provided by the LFG architecture is of considerable benefit in cross-language NLP applications. One possible immediate application is transfer-based machine translation. The GramLab project on automatic multilingual LFG acquisition has provided resources to automatically

parse text in languages including Chinese, French, Spanish, German, Japanese and English into f-structures. If these source language f-structures can be mapped into target language f-structures, a generator as proposed in this thesis can generate target language text from these f-structures. The overall performance of such a transfer-based machine translation system is highly dependent on the accuracy of the f-structure parser and f-structure-based generator (among others). In this sense, highly accurate parsers and generators will be vital components in transfer-based machine translation systems.

Appendix A

Feature Standardisation

This appendix lists the features and their possible values as used in the f-structure representation automatically derived from the Penn Chinese Treebank.

Atomic-Valued Features

NTYPE: common, proper, temporal

PRON-TYPE: person, reflexive, interrogative, demonstrative

NUMBER-TYPE: cardinal, ordinal

NUM: singular, plural

PERS: 1, 2, 3

GEND: male, female, nonhuman

ADJ-TYPE: appositive, beneficiary, condition, direction, extent, locative, manner, negative, purpose, temporal

ASPECT: 着, 了, 过

MOOD: 吗, 呢, 吧, 的, 了etc.

MSP: 而, 所, 以, 来, 去

DE: ±

DI: ±

DR: ±

VNV: ±

VNV-FORM: 不, 一

CLAUSE-TYPE: declarative, interrogative, exclamatory

PRECOORD-FORM: numbers and symbols

Grammatical Functions

SUBJ: Subjects.

OBJ: Direct objects and objects of certain prepositions.

OBJ2: Indirect objects.

OBL: OBL represents the general type of obliques. Also two special types are used in my f-structure representation:

OBL-AG: logic subject

OBL-LOC: obligatory location

COMP: Subordinate clauses that provide their own subjects.

XCOMP: Subordinate clauses whose subject is provided from elsewhere in the sentence.

DET: Determiners.

QUANT: Quantifiers composed of classifier and number in counting.

NUMBER: Numbers.

RESULT: Resultative complements.

COORD: Coordinates.

ADJUNCT: Adjuncts.

ADJ-REL: Adjuncts in relative clauses.

TOPIC: Topics.

Bibliography

Harald Baayen and Richard Sproat. 1996. Estimating lexical priors for low-frequency morphologically ambiguous forms. *Computational Linguistics*, 22(2):155–166.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 42–48. Saarbrücken, Germany.

Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of the 1st International Natural Language Generation Conference*, pages 1–8. Mitzpe Ramon, Israel.

John A. Bateman. 1997. Enabling technology for multilingual natural language generation: The KPML development environment. *Journal of Natural Language Engineering*, 3(1):15–55.

Richard Bellman. 1962. Dynamic programming treatment of the traveling salesman problem. *Journal of the ACM*, 9(1):61–63.

Anja Belz. 2007. Probabilistic generation of weather forecast texts. In *Proceedings of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies 2007*, pages 164–171. Rochester, NY, USA.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 313–320. Trento, Italy.

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the Chinese treebank. In *Proceedings of the 2nd Chinese Language Processing Workshop*, pages 1–6. Hong Kong, China.

Joan Bresnan. 2001. *Lexical-Functional Syntax*. Blackwell Textbooks in Linguistics. Wiley-Blackwell.

Michael Burke. 2006. *Automatic Annotation of the Penn-II Treebank with F-Structure Information*. Ph.D. thesis, School of Computing, Dublin City University, Dublin, Ireland.

Michael Burke, Olivia Lam, Rowena Chan, Aoife Cahill, Ruth O'Donovan, Adams Bodomo, Josef van Genabith, and Andy Way. 2004. Treebank-based acquisition of a Chinese lexical-functional grammar. In *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation*, pages 161–172. Tokyo, Japan.

Miriam Butt, Helge Dyvik, Tracy H. King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proceedings of the COLING-2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7. Taipei, TaiWan.

Miriam Butt, Tracy H. King, María-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. CSLI Publications, Stanford, CA, USA.

Aoife Cahill. 2004. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations*. Ph.D. thesis, School of Computing, Dublin City University, Dublin, Ireland.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd An-*

nual Meeting of the Association for Computational Linguistics, pages 319–326. Barcelona, Spain.

Aoife Cahill, Martin Forst, and Christian Rohrer. 2007. Stochastic realisation ranking for a free word order language. In *Proceedings of the 11th European Workshop on Natural Language Generation*, pages 17–24. Schloss Dagstuhl, Germany.

Aoife Cahill, Mairéad McCarthy, Josef van Genabith, and Andy Way. 2002. Automatic annotation of the Penn treebank with LFG f-structure information. In *Proceedings of the LREC Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*, pages 8–15. Las Palmas, Canary Islands, Spain.

Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1033–1040. Sydney, Australia.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 645–652. Barcelona, Spain.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 86–95. Toulouse, France.

Eugene Charniak. 1996. Tree-bank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University, Providence, RI, USA.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 598–603. Menlo Park, CA, USA.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139. Seattle, WA, USA.

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, page 456. Hong Kong, China.

Grzegorz Chrupała, Nicolas Stroppa, Josef van Genabith, and Georgiana Dinu. 2007. Better training for function labeling. In *Proceedings of the 2007 International Conference on Recent Advances in Natural Language Processing*. Borovets, Bulgaria.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Department of Computer & Information Science, University of Pennsylvania, Philadelphia, PA, USA.

Ann Copestake, Dan Fickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. 1995. Translation using minimal recursion semantics. In *Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 15–32. Leuven, Belgium.

Dick Crouch, Mary Dalrymple, Ronald M. Kaplan, Tracy H. King, John Maxwell, and Paula Newman. 2006. XLE documentation. Technical report, Palo Alto Research Center, CA, USA.

Richard Crouch, Ronald M. Kaplan, Tracy H. King, and Stefan Riezler. 2002. A comparison of evaluation metrics for a broad coverage parser. In *Proceedings of the LREC Workshop: Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems*, pages 67–74. Las Palmas, Canary Islands, Spain.

Mary Dalrymple. 2001. *Lexical-Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings*

of the 5th International Conference on Languae Resources and Evaluation, pages 449–454. Genoa, Italy.

David DeVault, David Traum, and Ron Artstein. 2008. Practical grammar-based NLG from examples. In *Proceedings of the 5th International Natural Language Generation Conference*, pages 77–85. Salt Fork, Ohio, USA.

Markus Dickinson and Detmar Meurers. 2003. Detecting inconsistencies in treebanks. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories*, pages 45–56. Växjö. Sweden.

Péter Dienes and Amit Dubey. 2003a. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language*, pages 33–40. Sapporo, Japan.

Péter Dienes and Amit Dubey. 2003b. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 431–438. Sapporo, Japan.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the 2nd International Conference on Human Language Technology Research*, pages 138–145. San Diego, California.

Michael Elhadad and Jacques Robin. 1996. An overview of SURGE: a reusable comprehensive syntactic realization component. Technical Report 96-03, Department of Mathematics and Computer Science, Ben Gurion University, Israel.

Ji Fang. 2006. *The Verb Copy Construction and the Post-Verbal Constraint in Chinese*. Ph.D. thesis, Department of Asian Languages, Stanford University, Stanford, CA, USA.

Ji Fang and Tracy H. King. 2007. An LFG Chinese grammar for machine use. In *Proceedings of the Grammar Engineering Across Frameworks 2007 Workshop*, pages 144–160. Stanford, CA, USA.

ZhiWei Fang, LiMin Du, and Yu ShuiYuan. 2006. 基于混合模板的汉英口语对话系统句子生成器(A Chinese sentence generator based on hybrid-template for spoken dialogue system). *中国科学院研究生院学报(Journal of the Graduate School of the Chinese Academy of Sciences)*, 1:23–30.

Katja Filippova and Michael Strube. 2007. Generating constituent order in German clauses. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 320–327. Prague, Czech Republic.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Journal of Natural Language Engineering*, 6:15–28.

Anette Frank. 2000. Automatic f-structure annotation of treebank trees. In *Proceedings of the 5th International Conference on Lexical Functional Grammar*, pages 139–160. Berkeley, CA, USA.

Ryan Gabbard, Mitchell Marcus, and Seth Kulick. 2006. Fully parsing the Penn treebank. In *Proceedings of the Conference on North American Chapter of the Association of Computational Linguistics*, pages 184–191. New York, USA.

Adongbieke Gulila. 2005. Chinese sentence generation by using English generation system. In *Proceedings of the International Conference on Chinese Computing*. Singapore.

Yuqing Guo, Josef van Genabith, and Haifeng Wang. 2007a. Treebank-based acquisition of LFG resources for Chinese. In *Proceedings of LFG07 Conference*, pages 214–232. Stanford, CA, USA.

Yuqing Guo, Josef van Genabith, and Haifeng Wang. 2008a. Dependency-based n-gram models for general purpose sentence realisation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 297–304. Manchester, UK.

Yuqing Guo, Haifeng Wang, and Josef van Genabith. 2007b. Recovering non-local dependencies for Chinese. In *Proceedings of the 2007 Joint Conference on Empiri-*

cal Methods in Natural Language Processing and Computational Natural Language Learning, pages 257–266. Prague, Czech Republic.

Yuqing Guo, Haifeng Wang, and Josef van Genabith. 2008b. Accurate and robust LFG-based generation for Chinese. In *Proceedings of the 5th Internationl Natural Language Generation Conference*, pages 86–94. Salt Fork, Ohio, USA.

One-Soon Her. 1991. *Grammatical Functions and Verb Subcategorization in Mandarin Chinese*. Crane Publishing, Taipei, Taiwan.

Derrick Higgins. 2003. A machine learning approach to the identification of WH gaps. In *Proceedings of the 10th Conference on European Chapter of the Association for Computational Linguistics*, pages 99–102. Budapest, Hungary.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, The University of Edinburgh, Edinburgh, UK.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of 3rd International Conference on Language Resources and Evaluation*, pages 1974–1981. Las Palmas, Canary Islands, Spain.

Deirdre Hogan, Conor Cafferkey, Aoife Cahill, and Josef van Genabith. 2007. Exploiting multi-word units in history-based probabilistic generation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 267–276. Prague, Czech Republic.

Chu-Ren Huang and Louis Mangione. 1985. A reanalysis of de: Adjuncts and subordinate clauses. In *Proceedings of West Coast Conference on Formal Linguistics IV*, pages 80–91. University of California, Los Angeles, USA.

Valentin Jijkoun. 2003. Finding non-local dependencies: beyond pattern matching. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 37–43. Sapporo, Japan.

Valentin Jijkoun and Maarten de Rijke. 2004. Enriching the output of a parser using memory-based learning. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Barcelona, Spain.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143. Philadelphia, PA, USA.

Aravind K. Joshi and Yves Schabes. 1992. Tree adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata and Languages*, pages 409–432. North-Holland, Netherlands.

Ronald M. Kaplan. 1995. *The formal architecture of Lexical-Functional Grammar*, pages 7–27. CSLI Publications, Stanford, CA, USA.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–282. MIT Press, Cambridge, MA, USA.

Ronald M. Kaplan and Jürgen Wedekind. 2000. LFG generation produces context-free ganguages. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 425–431. Saarbrücken, Germany.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 200–204. Santa Cruz, CA, USA.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430. Sapporo, Japan.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 170–177. Seattle, WA, USA.

Irene Langkilde. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 2nd International Conference on Natural Language Generation*, pages 17–24. New York, USA.

Irene Langkilde and Kevin Knight. 1998a. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 704–710. Montreal, Quebec, Canada.

Irene Langkilde and Kevin Knight. 1998b. The practical value of n-grams in derivation. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pages 248–255. New Brunswick, New Jersey.

Benoit Lavoie and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 265–268. Washington, DC, USA.

Roger Levy and Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 439–446. Sapporo, Japan.

Roger Levy and Christopher Manning. 2004. Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 327–334. Barcelona, Spain.

Tangqiu Li, Eric H. Nyberg, and Jaime G. Carbonell. 1996. Chinese sentence generation in a knowledge-based machine translation system. Technical Report CMU-CMT-96-148, Center for Machine Translation, Carnegie Mellon University, PA, USA.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference on Human Language Technology and the North American Chapter of the Association for Computational Linguistics*, pages 71–78. Edmonton, Alberta, Canada.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the International Joint Conference on AI*, pages 1420–1427. Montréal, Canada.

Peiqi Liu, Zengzhi Li, and Yinliang Zhao. 2005. Chinese sentences generating in network fault diagnosis expert system. *International Journal of Information Technology*, 11(12):1–13.

Tomasz Marciniak and Michael Strube. 2004. Classification-based generation using TAG. In *Proceedings of the 3rd International Conference on Natural Language Generation*, pages 100–109. Brockenhurst, UK.

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

Mairéad McCarthy. 2003. *Design and Evaluation of the Linguistic Basis of an Automatic F-Structure Annotation Algorithm for the Penn-II Treebank*. Master's thesis, School of Computing, Dublin City University, Dublin, Ireland.

Susan W. McRoy, Songsak Channarukul, and Syed S. Ali. 2000. YAG: a template-based generator for real-time systems. In *Proceedings of the 1st International Natural Language Generation Conference*, pages 264–267. Mitzpe Ramon, Israel.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. Suny Series in Linguistics. State University of New York Press, New York, USA.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*, pages 285–291. Borovets, Bulgaria.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn treebank. In *Proceedings of the 1st International Joint Conference on Natural Language Processing*, pages 684–693. Hainan Island, China.

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the 9th International Workshop on Parsing Technology*, pages 93–102. Vancouver, British Columbia.

Joakim Nivre. 2006. *Inductive Dependency Parsing*, volume 34 of *Text, Speech and Language Technology*. Springer.

Alice H. Oh and Alexander I. Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 27–32. Seattle, WA, USA.

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

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 433–440. Sydney, Australia.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies 2007*, pages 404–411. Rochester, New York.

Carl Pollard and Ivan Sag. 1994. *Head-driven Phrase Structure Grammar*. CSLI Publications, Stanford, CA, USA.

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 194–201. Seattle, WA, USA.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

Ehud Reiter, Somayajulu Sripada, and Roma Robertson. 2003. Acquiring correct knowledge for natural language generation. *Journal of Artificial Intelligence Research*, 18:491–516.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278. Philadelphia, PA, USA.

Stefan Riezler and John T. Maxwell. 2006. Grammatical machine translation. In *Proceedings of the Conference on Human Language Technology and the North American Chapter of the Association of Computational Linguistics*, pages 248–255. New York, NY, USA.

Eric Ringger, Michael Gamon, Martine Smets, Simon Corston-Oliver, and Robert C. Moore. 2003. Linguistically informed statistical models of constituent structure for ordering in sentence realization. Technical Report MSR-TR-2003-54, Microsoft Research, Microsoft Corporation, Redmond, WA, USA.

Louisa Sadler, Josef van Genabith, and Andy Way. 2000. Automatic f-structure annotation from the AP treebank. In *Proceedings of the 5th International Conference on Lexical Functional Grammar*, pages 226–243. Berkeley, CA, USA.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 162–168. Geneva, Switzerland.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 177–184. Sydney, Australia.

Mark Steedman. 1996. *Surface Structure and Interpretation*. MIT Press, Cambridge, MA, USA.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 341–351. Mexico City, Mexico.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of International Conference of Spoken Language Processing*, pages 901–904. Denver, Colorado, USA.

Kiyotaka Uchimoto, Masaki Murata, Qing Ma, Satoshi Sekine, and Hitoshi Isahara. 2000. Word order acquisition from corpora. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 871–877. Saarbrücken, Germany.

Josef van Genabith, Louisa Sadler, and Andy Way. 1999. Semi-automatic generation of f-structures from treebanks. In *Proceedings of the 4th International Conference on Lexical Functional Grammar*, pages 1–20. Manchester, UK.

Erik Velldal and Stephan Oepen. 2005. Maximum entropy models for realization ranking. In *Proceedings of the 10th Machine Translatin Summit*, pages 109–116. Phuket, Thailand.

Erik Velldal, Stephan Oepen, and Dan Flickinger. 2004. Paraphrasing treebanks for stochastic realization ranking. In *Proceedings of the 3rd Workshop on Treebanks and Linguistic Theories*. Tübingen, Germany.

Michael White. 2004. CCG chart realization from disjunctive inputs. In *Proceedings of the 3rd International Natural Language Generation Conference*, pages 182–191. Brockenhurst, UK.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *Proceedings of the MT Summit XI Workshop on Language Generation and Machine Translation*, pages 22–30. Copenhagen, Danmark.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium*, pages 398–403. Beijing, China.

Nianwen Xue. 2007. Tapping the implicit information for the PS to DS conversion of the Chinese treebank. In *Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories*, pages 189–200. Bergen, Norway.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Journal of Natural Language Engineering*, 11(2):207–238.

Huayan Zhong and Amanda J. Stent. 2005. Building surface realizers automatically from corpora. In *Proceedings of the Corpus Linguistics 2005 Workshop on Using Corpora for Natural Language Generation*, pages 49–54. Birmingham, UK.

Dexi Zhu. 1982. *语法讲义(Lectures on Syntax)*. 商务出版社(Commercial Press), Beijing, China.