Lexical Syntax for Statistical Machine Translation

Hany Hassan

BSc., MSc.

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University

Faculty of Engineering and Computing, School of Computing

Supervisors: Prof. Andy Way

Dr. Khalil Sima'an

January 2009

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:

(Candidate) ID No: 55154085

Date:

Abstract

Statistical Machine Translation (SMT) is by far the most dominant paradigm of Machine Translation. This can be justified by many reasons, such as accuracy, scalability, computational efficiency and fast adaptation to new languages and domains. However, current approaches of Phrase-based SMT lacks the capabilities of producing more grammatical translations and handling long-range reordering while maintaining the grammatical structure of the translation output. Recently, SMT researchers started to focus on extending Phrase-based SMT systems with syntactic knowledge; however, the previous techniques have limited capabilities due to introducing redundantly ambiguous syntactic structures and using decoders with limited language models, and with a high computational cost.

In this thesis, we extend Phrase-based SMT with lexical syntactic descriptions that localize global syntactic information on the word without introducing syntactic redundant ambiguity. We presente a novel model of Phrase-based SMT which integrates linguistic lexical descriptions —supertags— into the target language model and the target side of the translation model. We conduct extensive experiments in two language pairs, Arabic–English and German–English, which show significant improvements over the state-of-the-art Phrase-based SMT systems.

Moreover, we introduce a novel Incremental Dependency-based Syntactic Language Model (IDLM) based on wide-coverage CCG incremental parsing which we integrate into a direct translation SMT system. Our proposed approach is the first to integrate full dependency parsing in SMT systems with a very attractive computational cost since it deploys the linear decoders widely used in Phrase–based SMT systems. The experimental results show a good improvement over a top-ranked state-of-the-art system.

Acknowledgements

First of all, I am deeply grateful to Allah who gave me the confidence to pursue this work and complete my PhD studies.

I would like to express my sincere gratitude to my supervisors Andy Way and Khalil Sima'an. I would like to thank Andy Way for his constant support and encouraging. He gave me the chance and the arrangements to start this work, he gave me the freedom to pursue my research in various directions, and he was always encouraging and supporting. Thanks to you Andy. I would like to thank Khalil Sima'an for the fruitful and insightful discussions which I really enjoyed with him. Without his inspirational guidance and enthusiasm, this work would not have been as it is now.

I am very grateful to Salim Roukos for his support and encouraging, without his support this work would have take much longer time. Many thanks to Abe Ittycheriah for his assistance and support.

I want to express my gratitude to Ahmed Tantawy for inspiration, unlimited support and encouraging, without his support this work would not have started at all. I would also like to thank Ossama Emam and all the members of IBM Cairo-HLT team for support and understanding.

I would like to thank Mary Hearne for providing me with the template for editing this thesis. Thanks to Yanjun for his help while I am not at DCU.

I owe my parents much of what I have become. I thank them for their support and encouraging. I would also like to thank my kids Anas and Mohamed for providing me with joyful moments that keep me running

Last but definitely not least, this thesis would not have been possible without the support, understanding and patience of my soulmate, my wife Reham. Thanks to you Reham.

Contents

Abstract i

Acknowledgment ii

1 Introduction 1

1.1 Structure of the Thesis 6

1.2 Publications 7

2 State-of-the-art in Statistical Machine Translation 8

2.1 Introduction 8

2.2 MT Overview 9

2.3 The Noisy Channel Model 10

2.3.1 Word-Based Models 11

2.4 Phrase-Based Models 13

2.4.1 Overview 13

2.4.2 Phrase-based SMT Mathematical Model 14

2.4.3 Log-Linear Representation 15

2.4.4 Log-Linear Model Parameter Estimation 17

2.4.5 Phrase Extraction 18

2.4.6 Reordering 20

2.4.7 Phrase-based Decoder 21

2.5 Syntax support for Phrase-based SMT 23

2.5.1 Syntax Support via Reranking 23

2.5.2 Hierarchical Phrase-Based Translation 23

2.5.3 Syntactified Phrase-based MT (SPMT) 24
2.5.4 Syntax-Augmented Machine Translation 25
2.6 Direct Translation Models 25
2.6.1 Limitations of Log-Linear Phrase-based Model 25
2.6.2 Direct Translation Model 27
2.6.3 Direct Translation Model 2 (DTM2) 28
2.7 MT Evaluation Overview 30
2.8 Summary 31

3 Lexical Syntax Overview **33**
3.1 Syntax and Parsing 33
3.2 Lexicalized Grammars 34
3.3 Lexicalized Tree Adjoining Grammar 35
3.3.1 LTAG Supertagging 37
3.4 Combinatory Categorial Grammar 38
3.4.1 CCG Combinatory Operators 39
3.4.2 CCG Supertagging 42
3.5 Comparison between LTAG and CCG 43
3.6 Syntactic Language Models 44
3.6.1 Previous work 44
3.6.2 Lexical Syntax Language Models 45
3.7 Summary 46

4 Syntactic Phrase-based SMT: The Supertagging Approach **47**
4.1 Introduction 47
4.2 Syntax and Phrase based SMT 48
4.3 Supertagging for Phrase-based SMT 51
4.4 Why Supertagged Phrase-based SMT? 52
4.5 Our Approach: Supertagged Phrase-based SMT 55

4.5.1 A Supertag-Based SMT model 55
4.5.2 Language Models with a Grammaticality Factor 59
4.5.3 Supertagged Phrase-based Decoder 62
4.6 Experiments 62
4.6.1 Arabic–to–English 62
4.6.2 German–to–English 74
4.7 Conclusions and Open Questions 76

5 Incremental Dependency-based Language Modeling 79

5.1 Introduction 79
5.2 Incremental Dependency-based Language Model for MT 80
5.2.1 From Supertagged to Dependency-based Language Models 80
5.2.2 IDLM Overview 81
5.3 Related Work 84
5.3.1 Syntax-based Language Models 84
5.3.2 Incremental Parsing: Related Work 85
5.4 Incremental Parsing for CCG 87
5.4.1 Merits of CCG for Incrementality? 89
5.4.2 CCG Dependency Structure 92
5.5 Transforming the CCGbank into left-to-right Derivations 92
5.5.1 Incremental Combinatory Operators 96
5.5.2 Coordination 99
5.5.3 Apposition and Interruption 100
5.5.4 WH-movement 101
5.6 Implementation Details of the Incremental Parser 102
5.6.1 Supertagger and Operator tagger 102
5.6.2 Parse-State Realizer 103
5.7 Experiments and Results 104
5.7.1 Supertagging Results 105

5.7.2 Operator Tagging Results 106
5.7.3 Dependency Results 106
5.7.4 Cascaded vs. Joint Approach 107
5.7.5 Effect of Lookahead 107
5.7.6 Examples 108
5.8 Discussion 113

6 Dependency-based SMT **115**
6.1 Introduction 115
6.2 Dependency-based Language Model for SMT 116
6.2.1 IDLM Representation for SMT 116
6.2.2 Linear-time, Incremental Parsing Decoder 117
6.3 Related Work 119
6.4 Difficulties of incorporating IDLM into Phrase-based SMT 120
6.5 Dependency-based Direct Translation Model (DDTM) 120
6.5.1 Model Overview 120
6.5.2 DDTM Features 122
6.5.3 DDTM Decoder 122
6.6 Experiments 127
6.6.1 Results 128
6.7 Results Analysis 130
6.8 Conclusion 132

7 Conclusions **134**
7.1 Contribution of the Thesis 134
7.2 Future Research Avenues 138
7.3 Closing Remark 139

Bibliography **140**

List of Tables

4.1 Baseline systems’ BLEU scores 64

4.2 LTAG and CCG systems on small data 65

4.3 Comparing the effect of supertags and POS tags 65

4.4 The effect of supertagged components 66

4.5 Performance on large training data 67

4.6 CCG with grammaticality factor (CCG-LARGE-GRAM) 67

4.7 Sentence average BLEU score for CCG, LTAG and oracle with both . . . 68

4.8 Sentence average BLEU score for CCG, LTAG and oracle with both . . . 68

4.9 How CCG improves over baseline 70

4.10 How Baseline improves over CCG 73

4.11 CCG Supertags System for German–English 74

5.1 Operators’ Utilization. 97

5.2 Supertagger, Operator tagger and Dependency results (F-Score) of all systems. 105

6.1 DDTM systems with associated features 128

6.2 DDTM Results with various features. 129

List of Figures

2.1 Various SMT models projected on the three-dimensional model space . . 10

2.2 SMT system: training and runtime (decoding) 11

2.3 Phrase–based SMT. The Arabic source is segmented with arbitrary phrase boundaries then translated to English phrases which are reordered as needed. 14

2.4 Extracting Phrase Alignments from Word Alignments (from (Groves & Way, 2005), p.310) 19

2.5 All possible source segmentations with all possible target translations (from (Koehn, 2004)) 21

2.6 Expanding the decoder hypothesis with possible translations (from (Koehn, 2004)) 22

2.7 Phrase structures in DTM2. X represents a variable in the target phrase. From (Ittycheriah and Roukos, 2007) 29

3.1 CFG production rules. 34

3.2 LTAG elementary trees for the sentence with the parse tree resulting from combining the elementary trees (from (Hassan et al., 2008a)) 36

3.3 LTAG derivation tree that produced the parse tree in Figure 3.2. Initial trees are inserted using substitution and auxiliary trees are inserted by adjunctions. 37

3.4 CCG Supertags and the derivation combining the supertags into a parse-tree. 39

4.1 Arabic and English aligned phrase pairs with the English constituent structure (from (Hassan et al., 2008a)). 49

4.2 Mismatch between PBSMT phrases and tree structures (from (Hassan et al., 2008a)). . . . 50
4.3 Supertagged PBSMT system (from (Hassan et al., 2008a)) 60
4.4 A grammatical violation example. 62
4.5 Conflict of CCG and LTAG when combined 69
4.6 Improved Reordering in the CCG system 70
4.7 Overcoming missing verbs in the CCG system 70
4.8 Baseline system omits verbs while supertags system can produce them . . 71
4.9 Improved Verb Forms in the CCG system 72
4.10 Improved translation in general in the CCG system 72
4.11 Restored negation in the CCG system 72
4.12 Better syntactic modelling in the CCG system 72
4.13 Improved performance of the CCG system for German–English. 76
4.14 Overcoming missing verbs in the CCG system for German–English 76

5.1 Illustration of the incremental parser representation and the associated intermediate dependency graphs at each state. 88
5.2 A sentence and a possible supertag sequence, operator sequence and state sequence. NOP: No Operation; TRFC: Type-Raise Forward Composition; FA: Forward Application. 89
5.3 Incremental Parser: a: Transformation & Training phase, b: Parsing runtime phase. 90
5.4 CCG dependency structure. 92
5.5 Illustration of the CCGbank transformation process into incremental derivations (from (Hassan et al., 2008b)). 94
5.6 Coordination Handling. 99
5.7 VP Coordination Handling. 100
5.8 Apposition Handling. 101
5.9 WH-movement Handling. 102

5.10 Illustration of the operation of the incremental parse-state realizer and the associated intermediate dependency graphs at each state. 104

5.11 Example1: Incremental Parsing. 110

5.12 Example 2: Incremental Parsing. 111

5.13 Example 3: Incremental Parsing. 112

6.1 DDTM Decoder: each hypothesis has a parse state and a partial dependency structure. 125

6.2 Partial decoding graph for a sentence: We show hypothesis paths along with supertags and operators associated with each word and the parse-state category associated with each decoding state. 126

6.3 DDTM opts for inserting verbs and breaking long noun phrases with prepositions. 131

6.4 DDTM provides better syntatctic structure with more concise translation. 131

6.5 Example: Long range reordering and the need for lexicalization. 132

6.6 Better long-range reordering 132

Chapter 1

Introduction

Statistical Machine Translation (SMT) is by far the most dominant paradigm in machine translation today. This can be justified by many reasons, such as accuracy, scalability, computational efficiency and fast adaptation to new languages and domains. Seeking better translation quality, SMT has evolved from the IBM word-based models (Brown et al., 1988, 1990) to phrase-based models (Zens et al., 2002; Koehn et al., 2003; Tillmann and Xia, 2003) . However, Phrase-based SMT lacks the capability of producing more grammatical translations and handling long-range reordering while maintaining the grammatical structure of the translation output. The main objective of this thesis is to produce more fluent MT output from Phrase-based SMT by integrating syntactic structures into the system. Syntax can help Phrase-based SMT systems to produce well-formed translation output by the use of syntactically-guided translation models, language models and reordering techniques.

Recently, SMT researchers started to focus on extending Phrase-based SMT systems with syntactic knowledge; however, early attempts caused system performance to deteriorate (Koehn et al., 2003). The most recent successful enrichments of Phrase-based SMT with hierarchical structures are (Chiang, 2005; Marcu et al., 2006; Zollmann and Venugopal, 2006).

We argue that these previous techniques have limited capabilities due to three major drawbacks. Firstly, these approaches either employ non-linguistically motivated syntax to capture hierarchical reordering phenomena (Chiang, 2005) or extend the phrase translation table with redundantly ambiguous syntactic structures over phrase pairs (Marcu et al., 2006; Zollmann and Venugopal, 2006). Secondly, they compromise the computational efficiency of the phrase-based system since they depart from the computationally efficient linear decoders to the more computationally costly chart-based decoders. Thirdly, they limit the scalability of the system by their limited capability to handle high-order language models which have proved to be pivotal to the accuracy of Phrase-based systems.

In this thesis, we study the possibility of extending Phrase-based SMT systems with syntactic structures to provide more grammatical translations and better reordering without compromising the advantages of such systems. This leads us to our first research question *(RQ1)*:

RQ1: What is the grammatical representation that can best fit with Phrase-based SMT while not introducing redundantly ambiguous syntactic structures?

Extending Phrase-based SMT systems with linguistically motivated syntax represents a major difficulty due to the mismatch between the notion of a ‘phrase’ in Phrase-based SMT and the notion of a syntactic constituent in traditional linguistics. The problem is that the phrases in Phrase-based SMT systems are identified with regard to word alignment probabilities and thus do not need to follow any linguistic constraints. Due to such mismatches, it is not directly clear how the SMT notion of a phrase may be extended with a tree structure without introducing redundant ambiguity. For a non-constituent phrase, a tree structure representation directly introduces redundant ambiguity; multiple, alternative subtrees will be associated with the same phrase, whereas they are merely minor variants of each other, differing only in subgraphs that denote very specific contexts of the phrase.

In this thesis, we explore the possibility of extending Phrase-based SMT with lexicon-driven approaches to linguistic syntax, namely Lexicalized Tree-Adjoining Grammar (Joshi and Schabes, 1991) and Combinatory Categorial Grammar (Steedman, 2000). In these approaches, each word is associated with a number of lexical entries which consist of syntactic constructs —supertags (Bangalore and Joshi, 1999)— that describe such lexical information as its subcategorization information and the hierarchy of phrase categories that the word projects upwards in the parse-tree. Thus, these lexical syntactic descriptions localize global syntactic information on the word level; therefore, they can be assigned to every word in a phrase without introducing much redundant ambiguity.

If we have an efficient syntactic representation that fits well with Phrase-based SMT, the question arises as to how to incorporate this into Phrase-based SMT. This is our second research question *RQ2*.

RQ2: How can lexical syntax descriptions be incorporated into Phrase-based SMT while maintaining its advantages? If this can be done, does it help in providing better and more grammatical translations?

We address this problem by presenting a novel model of Phrase-based SMT which integrates linguistic lexical descriptions —supertags— into the target language model and the target side of the translation model. We examine whether lexical syntactic information proves useful or not. We carry out extensive experiments on small and very large training and test sets for Arabic–English and German–English translation to examine the usage of LTAG and CCG supertags in different conditions. We compare the effect of CCG and LTAG with different data sizes. We examine whether the improvement provided by our approach will be sustained with very large amounts of training data, or whether large amounts of training data would bridge the performance gap with our system that incorporates syntactic knowledge.

We conduct an in-depth manual analysis of the system performance. We show that a very wide range of improvements are brought about by the use of a supertags-based system, including improved reordering, overcoming the tendency of SMT systems to omit verbs, improved verbal constructions, proper handling of negation, and better syntactic modeling in general. We show that supertagged Phrase-based SMT provides sustained improvements with various data sets and languages.

The encouraging results of our proposed supertagged Phrase-based SMT approach leads to our next research question *RQ3*:

RQ3: Does Phrase-based SMT need more syntactic knowledge or is our supertagged approach sufficient for providing syntactic structures to enable more grammatical translations and better reordering?

The supertagged language model replaces the set of Combinatory Operators with an *n*-gram language model over the sequence of supertags thus providing *'almost parsing'* (Bangalore and Joshi, 1999). Originally, *'almost parsing'* was proposed for the handling of monolingual strings, where the given sequence of words already constructs some presumed syntactic structure. In the bilingual (machine translation) case, the sequence of candidate target words might not construct a valid syntactic structure nor a compelling sequence of associated supertags; therefore, achieving *'almost parsing'* by deploying a supertagged *n*-gram language model on the huge space of hypotheses, representing the candidate translations, is more challenging in the machine translation case than in the monolingual parsing case.

We argue that the MT case needs a more sophisticated mechanism that can satisfy three important aspects. First, it needs to efficiently support long-range dependencies and construct full parse structures such that it would enable the MT system to distinguish between different translation candidates based on their role in constructing the parse structure and satisfying the syntactic dependencies. Second, as is widely known, Phrase-based SMT systems produce the translation candidates incrementally by processing source words from left-to-right in a Markov fashion; therefore, this mechanism should work in an incremental manner. Third, the mechanism should be computationally efficient such that it can be integrated into large-scale Phrase-based SMT systems.

While incrementality is crucial for integrating syntax into SMT decoders, it is not necessary for reranking of SMT output. However, reranking SMT output usually does not provide good improvement due to the fact that SMT decoders deploy many strategies that may prune good candidates earlier in the translation process such that better translations may not even be part of a very large n-best candidate set.

Having identified the need for an incremental dependency parser that could fit with Phrase-based SMT, this directly leads to our next research question *RQ4*:

RQ4: Can lexical syntax provide more syntactic knowledge for Phrase-based SMT through incremental dependency parsing capabilities that match the nature of Phrase-based SMT?

We address this problem by introducing a novel incremental dependency-based language model parser. To develop this incremental dependency-based parser, we create an incremental version of CCGBank (Hockenmaier and Steedman, 2007) and examine its usefulness for developing a fully incremental parser. We conduct extensive experiments to examine the proposed incremental parser with regard to various incrementality issues as well as the accuracy of the parser.

We demonstrate that the proposed parser can be used to develop an incremental dependency-based language model (IDLM). We show that this IDLM is deterministic in that it maintains a limited number of parsing decisions at each state which makes it very efficient for integration into large-scale Phrase-based SMT systems. Furthermore, we show that it can naturally handle non-constituent constructions, being based on CCG. Finally, we show that the parser always seeks fully connected structures and can support long-range dependencies and a number of interesting syntactic phenomena in a fully incremental left-to-right fashion. However, it remains to be seen whether we can incorporate a linear though sophisticated incremental parser into Phrase-based SMT while maintaining scalability and computational efficiency. This is the subject of our last research question.

RQ5: Is it possible to incorporate full incremental dependency parsing into SMT while maintaining SMT scalability and computationally efficient linear decoding?

One major difficulty in extending SMT systems with a sophisticated incremental dependency-based language model is the need for a well-formalized model that can accommodate the capabilities of a conventional phrase-based system and the incremental dependency parsing without compromising any of their advantages, while at the same time maintaining a reasonable decoding space. We address this problem by proposing an extension of a discriminative phrase-based SMT model (DTM2) (Ittycheriah and Roukos, 2007) where we represent the incremental parser efficiently as a large number of features. We examine the capabilities of the proposed model and show that it can provide improvements over top-ranked SMT systems. We conduct a detailed analysis of the system output to obtain a deeper insight into the system's performance.

In this thesis, we explore the possibility of improving the translation quality of Phrase-based SMT systems by incorporating syntactic structures in the target side while dealing with the non-constituent phrases. Furthermore, we will explore the possibility of incorporating syntactic structures into Phrase-based SMT systems without compromising the computational efficiency of the linear decoders or the large language models capabilities. We will explore various levels of syntactic integration that can provide more syntactic knowledge to SMT.

1.1 Structure of the Thesis

The remainder of this thesis is organized as follows:

- Chapter 2 introduces an overview of the state-of-the-art in SMT with the noisy channel model, log-linear phrase-based models and direct translation models. The chapter also reviews the previous work on incorporating syntax into SMT.

- Chapter 3 introduces an overview of lexical syntax and lexicalized grammars.

- Chapter 4 introduces the concept of lexical syntax for SMT and explores how we incorporate supertagged translation model and supertagged *n*-gram language model into Phrase-based SMT. Extensive experiments are reported for two pairs of languages.

- Chapter 5 introduces a novel incremental dependency-based language model based on an incremental version of CCG, and introduces experiments for evaluating the proposed parser.

- Chapter 6, discusses the integration of our incremental dependency-based language model into SMT

- Finally Chapter 7 concludes and discusses avenues for future work.

1.2 Publications

A number of publications were based on the work in this thesis:

- (Hassan et al., 2006) which is entitled "Syntactic Phrase-Based Statistical Machine Translation" and was published in the Proceedings of the 2006 IEEE Workshop on Spoken Language Technology.

- (Hassan et al., 2007a) entitled "MaTrEx: the DCU Machine Translation System for IWSLT 2007" was published in proceedings of the International Workshop on Spoken Language Translation

- (Hassan et al., 2007b) entitled "Integrating Supertags into Phrase based Statistical Machine Translation" was published in the Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07).

- (Hassan et al., 2008a) entitled "Syntactically Lexicalized Phrase-based Statistical Translation" was published in the IEEE Transactions on Audio, Speech and Language Processing journal.

- (Hassan et al., 2008b) entitled "A syntactic language model based on incremental CCG parsing" was published in the Proceedings of the 2008 IEEE International Workshop on Spoken Language Technology.

Chapter 2

State-of-the-art in Statistical Machine Translation

2.1 Introduction

Machine translation (MT) is the process of using computers to translate text from one natural language into another. Statistical machine translation (SMT) is an MT paradigm where translations are generated using statistical models whose parameters are derived from the analysis of bilingual and monolingual text corpora. The idea of performing SMT by information theory methods was proposed a long time ago in (Weaver, 1949), who proposed that the statistical techniques from information theory and cryptography might make it possible to use computers to translate text from one natural language to another.

Four decades later, in the late 1980's, a group of IBM researchers revisited the idea of using statistical techniques for translation. They were encouraged by the increase in computing power, the availability of large-scale parallel corpora and the lack of progress by other methods. (Brown et al., 1988, 1990) formulated the MT problem as a noisy channel model, which has led to the rise of SMT as we experience today. SMT is now by far the most dominant paradigm of MT for many reasons, such as accuracy, scalability and fast adaptation to new languages and domains. In this chapter, we introduce an overview of the state-of-the-art in SMT.

2.2 MT Overview

The classical architecture of MT systems follows the Vauquois Triangle (Vauquois, 1968). This representation proposes that there are three main paradigms for MT, namely Direct, Transfer and Interlingua. This classical architecture helped in understanding various processes that might be used in performing MT; however, MT systems rarely adhere to this claimed theoretical framework due to the compromised solutions assumed during systems development.

A more recent representation was proposed by (Wu, 2005), in which he presented a three-dimensional MT model space that focused on the components deployed to achieve the translation rather than on the process of performing the translation. (Wu, 2005)'s 3D-representation consists of three dimensions: statistical versus logical, compositional versus lexical, and example-based versus schema-based. (Wu, 2005) defines SMT as an MT system that makes nontrivial use of statistics and probability while the logical system makes extensive use of logical rules. Compositional MT uses compositional transfer transduction rules while lexical MT uses lexical transfer without compositional rules. Finally, Example-based MT uses a large library of examples at translation runtime while Schema-based MT uses abstract schemata during runtime.

Figure 2.1 shows the projection of different SMT systems in this three-dimensional model. Word-based SMT models represent the statistical and lexical combination, while Phrase-based SMT systems deploy more collocational information and therefore move away from the lexical towards the compositional dimension. As more syntactic knowledge is added into Phrase-based SMT, the system is pushed further into the compositional dimension. In the next sections we will review Word-based models and Phrase-based models.

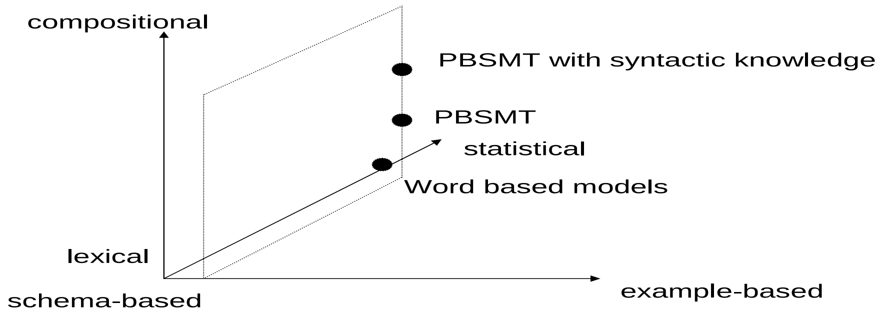


Figure 2.1: Various SMT models projected on the three-dimensional model space

2.3 The Noisy Channel Model

(Brown et al., 1988, 1990) proposed that the problem of MT can be handled as a noisy channel model. A target sentence T is transferred to a source sentence S when going through a noisy channel. If this noisy channel could be modelled, then translation from S to T could be achieved. The machine translation decoder reverses the noisy channel by reproducing the target sentence T from the source sentence S. As shown in Eqn (2.1), the source channel model is composed of two components: the translation model and the language model.

$$\begin{aligned} T^* &= \arg\max_T P(T|S) = \arg\max_T P(S \mid T)P(T)/P(S) \\ &\approx \arg\max_T \overbrace{P(S \mid T)}^{TranslationModel} \overbrace{P(T)}^{LanguageModel} \end{aligned} \tag{2.1}$$

Figure 2.2 demonstrates the SMT system in the training and decoding phases. At training time, the system uses a parallel corpus to estimate the translation model probabilities and a monolingual corpus to estimate the target language model probabilities. At decod-

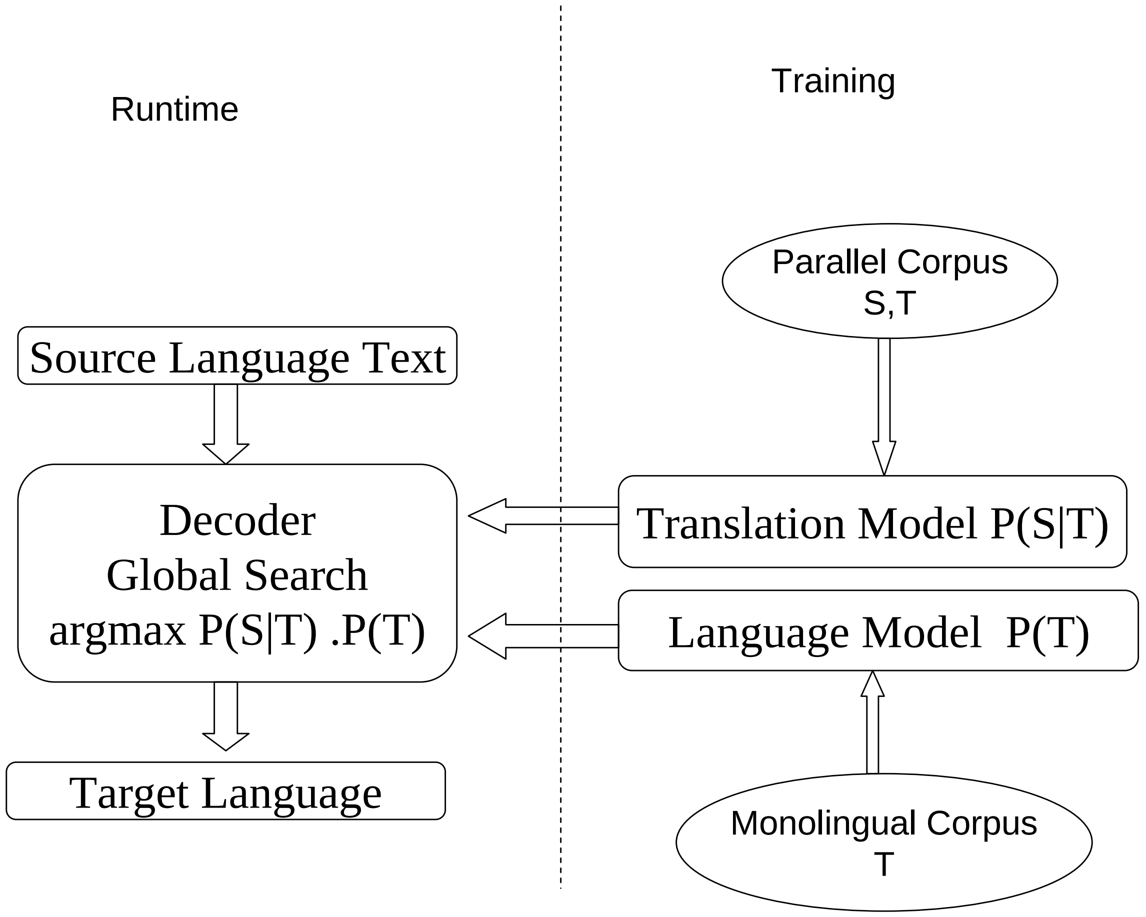


Figure 2.2: SMT system: training and runtime (decoding)

ing time, the two probabilistic components are utilized within a global search technique to find the best translation for a given source sentence. The translation model can take various forms such as word-based, phrase-based and syntax-based. The language model can be an *n*-gram language model, syntax-based language model or any other model that measures how fluent the target language output is. In the following sections we will review two forms of translation models: word-based and phrase-based.

2.3.1 Word-Based Models

In word-based translation models, the translation model in Eqn (2.1) is simply a word-to-word translation probability which has been estimated from word alignments that represent a mapping between source and target words in a parallel sentence pair. Word alignment is crucial for SMT as the accuracy of the translation component is highly dependent on it. As shown in Eqn (2.2), the translation probability of a source sentence S and a target sentence T is the sum over all possible alignments A between the source and the target sentences. (Brown et al., 1988, 1990) proposed five alignment models with different complexity known as IBM Model1 – Model5.

$$P(S|T) = \sum_{A} P(S, A|\, T) \tag{2.2}$$

In all IBM alignment models a source word can be linked to exactly one target word, thus these alignment models do not allow many-to-one or many-to-many alignments. IBM Model1 is the simplest among these models which aim to learn the word (lexical) translation model using the alignment links. If we already know the alignment links, we can estimate the lexical translation model by collecting counts, as stated in Eqn (2.2), and performing maximum likelihood estimation. On the other hand, if we have the translation model we can assume the most likely alignment links. The problem is that we do not have either of them. This is a well-known problem, when a model is being estimated from incomplete data where there are hidden variables in the model. In the case of Model1, we are trying to estimate the translation probabilities while the alignment links are the hidden variables in this problem. IBM Model1 uses the Expectation Maximization (EM) algorithm (Dempster et al., 1977) to solve this problem.

EM is applied in two steps: the expectation step (E-step) and the maximization step (M-step). First, IBM Model1 initializes all the translation probabilities with a uniform probability distribution, i.e. each source word can be the translation of each target word with the same uniform initial probability. Then the E-step is applied by computing the expected counts for the translation model based on summing over the alignments. In the M-step, the maximum likelihood estimate of the translation model is computed from these counts. The E-step and M-step are repeated iteratively until convergence.

IBM Model1 is the simplest among the five models since it only models the lexical translation probability. IBM Model2 models the word deletion probability as well by introducing a null word and estimating the probability of words being aligned to the null word. IBM Model3, in addition to modeling lexical translation and word deletion probability, models the fertility of each target word t_i, indicating the number of source words it may generate. IBM Model4 goes a step further and models the relative positions of the source and the target words for reordering. Model4 can generate a source sentence S from a target sentence $T = t_1, t_2....t_I$ as follows:

1. Each target word t_i has a particular fertility indicating the number of words it may generate and thus the length of S is the summation of the fertilities of all target words.

2. Each target word produces a number of source words according to its fertility with a translation probability.

3. The source words are reordered.

As we described for IBM Model1, EM is used to estimate the parameters of all other models. IBM Model4 is the most widely used model for word alignment. For detailed information about the mathematical formalization of IBM models, the reader is referred to (Brown et al., 1993). IBM models are implemented in the widely used toolkit GIZA++[1] (Och and Ney, 2003).

2.4 Phrase-Based Models

2.4.1 Overview

Word-based SMT models have a major disadvantage, namely that they do not use any contextual information for estimating the translation probability. If the translation unit is larger than a single word, the contextual effects will be captured and will help to produce better translations; moreover this should help with local reordering of words such as noun-adjective reordering between different languages. Phrase-based SMT has been proposed

[1]http://www.fjoch.com/GIZA++.html

to overcome these problems where the unit of translation is any sequence of adjacent words. As shown in Figure 2.3, Phrase-based SMT system starts by segmenting the source sentence into phrases with arbitrary boundaries then translates the source phrases into target phrases and finally performs reordering if applicable.

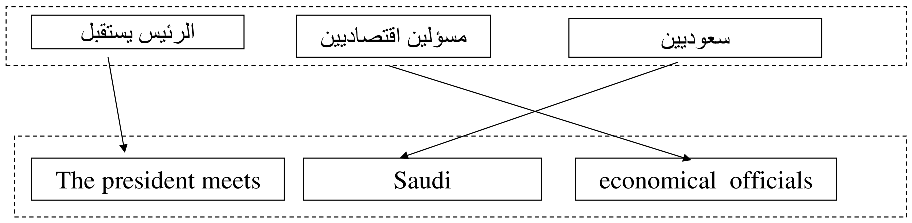


Figure 2.3: Phrase–based SMT. The Arabic source is segmented with arbitrary phrase boundaries then translated to English phrases which are reordered as needed.

The phrases of Phrase-based SMT are not linguistically motivated and do not necessarily relate to any constituent phrase structure. In fact, these phrases are called blocks or clumps in some literature (Tillmann and Xia, 2003); however we will use the term ’phrase’ throughout this thesis, while it should be clear that we mean the phrase as a sequence of words unless otherwise stated.

The current paradigm of Phrase-based SMT was proposed by different research groups (Zens et al., 2002; Tillmann and Xia, 2003; Koehn et al., 2003); however, there are more similarities than differences between the various approaches. In this section, we will review the Phrase-based SMT approach focusing on the commonly used techniques in the research community.

2.4.2 Phrase-based SMT Mathematical Model

Let s and t be an aligned pair of source and target sentences respectively. As is usually done in Phrase-based SMT, we assume a set of segmentations of s and t into phrase pairs.

We denote with $\sigma_{s,t}$ a segmentation of s and t. With every segmentation $\sigma_{s,t}$, there is a set of pairs of positions $O(s,t,\sigma)$ that retains the original ordering in s and t of the individual phrase pairs in $\sigma_{s,t}$. We will write O_s and O_t as the set of source and target positions in the pairs in $O(s,t,\sigma)$ respectively.

A given segmentation σ_x and ordering O_x define a derivation of sentence x in the following sense: the sentence x can be obtained by concatenating the phrases in ϕ_x according to the order O_x.

$$\begin{aligned} t^* &= \arg\max_t P(t|s) = \arg\max_t P(s \mid t)P(t) \\ &= \arg\max_t \sum_{\sigma,O} \overbrace{P(\phi_s \mid \phi_t)}^{TM} \overbrace{P(O_s \mid O_t)}^{distortion} \overbrace{P_{lm}(t)}^{LM} \\ &\approx \arg\max_{\sigma,O,\phi_t} \overbrace{P(\phi_s \mid \phi_t)}^{TM} \overbrace{P(O_s \mid O_t)}^{distortion} \overbrace{P_{lm}(t)}^{LM} \end{aligned} \tag{2.3}$$

In (2.3), $P_{lm}(t)$ is the target language model (LM) over word sequences, $P(O_s|O_t)$ represents the conditional reordering/distortion probability, and $P(\phi_s|\phi_t)$ stands for a probabilistic translation model from target language bags of phrases to source language bags of phrases under the segmentation $\sigma_{s,t}$ into a bag of phrase pairs. As shown in (2.3), the sum over segmentations is disregarded for the efficiency of optimization over target sentence and segmentation pairs.

Instead of the original formulation of the translation problem as a noisy-channel model, Phrase-based SMT employs a log-linear interpolation over a set of features as will be discussed next.

2.4.3 Log-Linear Representation

As described in the previous section, Phrase-based SMT consists of three probabilistic components: a phrase translation model (TM), reordering (distortion) model and the language model (LM). These components have resulted from applying the noisy channel representation on the Phrase-based system. Motivated by adding more components in a more flexible framework, (Och and Ney, 2002) proposed a log-linear representation for Phrase-based SMT. (Och and Ney, 2002) proposed a simplification of the direct translation model proposed in (Papineni et al., 1997, 1998) which uses Maximum Entropy (Berger et al., 1996) as a framework for utilizing many feature functions to model the direct translation probability $P(T|S)$.

(Och and Ney, 2002) represented formula (2.3) as a log-linear model interpolating a set of feature functions as in (2.4):

$$t^* = \arg\max_{t,\sigma} \prod_{f \in F} H_f(s,t,\sigma)^{\lambda_f} \tag{2.4}$$

The set F is a finite set of features and λ_f are the interpolation weights over feature functions H_f of the aligned source-target sentence pairs. The set of different features F employed in this approach consists of the following:

- lm: An *n*-gram language model over target sequences $H_{lm}(s,t,\sigma) = P(t) = \prod_i P(t_i|t_{i-4}^{i-1})$, where the language model probabilities are trained over a large monolingual corpus by Maximum-Likelihood estimation with an appropriate smoothing technique (Goodman, 2001).

- ϕ, r_ϕ: A source-target translation table is obtained from a word-aligned parallel corpus using phrase extraction heuristics (cf. Section 2.4.5). For every possible segmentation σ of the sentence pair $\langle s,t \rangle$, two feature weights are employed, namely $H_\phi(s,t,\sigma) = P(\phi_s \mid \phi_t)$ and its reverse $Hr_\phi(s,t,\sigma) = P(\phi_t \mid \phi_s)$.

  The phrase translation probability distribution is estimated by the relative frequency of a phrase pair in the multiset of phrase pairs obtained from the parallel corpus as in (2.5):

$$\hat{P}(s|t) = \frac{count(s,t)}{\sum_s count(s,t)} \tag{2.5}$$

Here $count()$ denotes the frequency count in the multiset of phrase pairs obtained from the aligned parallel training corpus.

- lex: For every phrase pair $\langle s_i, t_i \rangle$ (see feature ϕ), the system employs a model $H_{lex}(s,t,\sigma)$ based on estimates of $P_{lex}(s_i|t_i)$ (and the reverse direction) using lexical weights (word-to-word) as described in (Koehn et al., 2003). These weights provide a measure of the translation relations in the phrase pair using a lexicon of word-to-word translations obtained during the alignment phase,
- o: A phrase reordering model $H_o(s,t,\sigma)$ (cf. Section 2.4.6).
- x: The standard word/phrase penalty ($H_x(s,t,\sigma) = \exp^{-|t|}$) which allows for control over the length of the target sentence t.

2.4.4 Log-Linear Model Parameter Estimation

The parameters of each component of the log-linear model components are estimated independently. For example, the phrase translation probabilities are estimated from a bilingual corpus while the language model probabilities are estimated usually from a much larger monolingual corpus. The various components are interpolated in the log-linear framework by a set of parameters following the Maximum Entropy approach as shown in Eqn (2.4).

In the Maximum Entropy framework, each feature is associated with a weight. These weights can be estimated using iterative search methods to find a single optimal solution under the maximum entropy principle; however, this is a computationally expensive process. Therefore, (Och, 2003) proposed an approximation technique called Minimum Error Rate Training (MERT) to estimate the model parameters for a small number of features, which will be discussed in the next section. An error function that corresponds to the translation accuracy (Section 2.7) is defined and MERT estimates the log-linear model parameters such that this error function is minimized using the n-best output of the MT system.

MERT proceeds as follows:

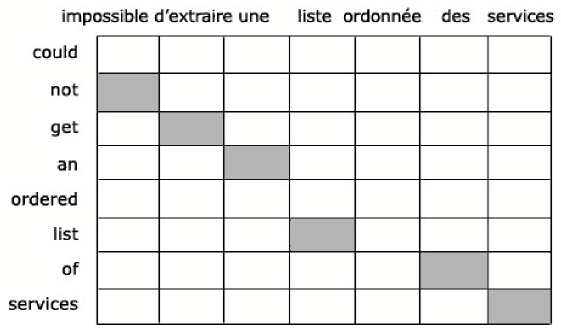
- Initialize all parameters with random values.
- Produce the n-best translations using the current parameter set.
- Compute the error function using reference translations.
- Optimize each parameter to minimize the error function while fixing all other parameters.
- Iterate over all parameters.

MERT provides a simple and efficient method to estimate the model parameters; however, it can only handle a small number of parameters (in the order of ten (Ittycheriah and Roukos, 2007)); when the number of parameters increases there is no guarantee that MERT is able to find the most suitable parameter combination.

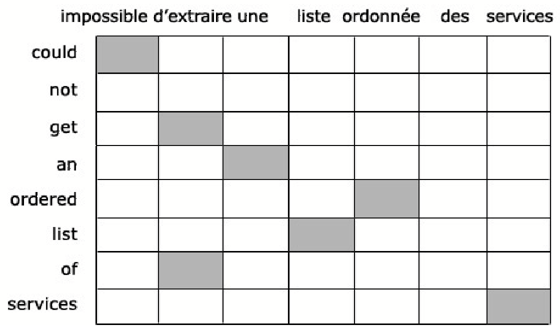
2.4.5 Phrase Extraction

IBM word alignment models provide word-to-word mapping where a source word can be aligned to exactly one target word. These alignment models do not allow for many-to-one or many-to-many alignments and so the alignments are asymmetric, i.e. the links of the alignment are not the same if the source and target language are swapped. As shown in Figures 2.4-a and 2.4-b, the alignment links are different when the languages are swapped.

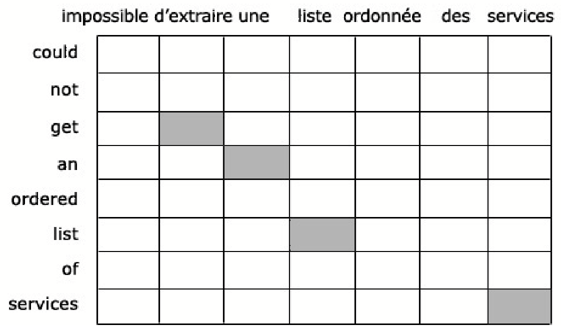
(Och and Ney, 2003) proposed an approach for extracting phrase mappings based on producing symmetrized alignments from word-based alignments and then using some heuristics to extract phrase pairs. First, alignments in both directions (target-source and source-target) are produced as shown in Figure 2.4-a and Figure 2.4-b respectively. Both



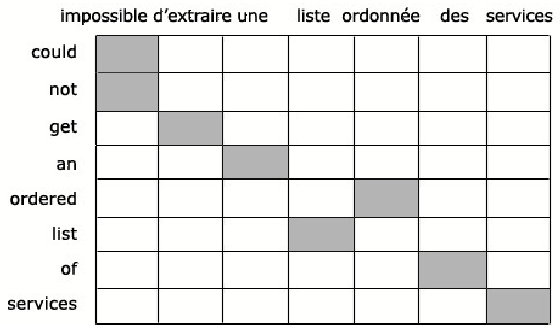
(a) English–French alignment



(b) French–English alignment



(c) Intersection of alignments



(d) Intersection extended to union

Figure 2.4: Extracting Phrase Alignments from Word Alignments (from (Groves & Way, 2005), p.310)

alignments are intersected to produce a high precision alignment as shown in Figure 2.4-c. The union of the two alignments is used to extend the intersection with more alignment points using some heuristics such as GROW-DIAGONAL which examines all the neighbouring alignment points of the intersections. If the neighbouring words are not in the intersection and if both their source and target words are in the union, then the alignments are extended with the union words. Finally, phrase pairs are extracted from those extended alignments as shown in Figure 2.4-d.

2.4.6 Reordering

Reordering defines how far the target phrase should move during translation. Generally, the reordering models penalize any movement in the target translation away from the corresponding source position and depend on the language model to judge how good this movement is. The basic reordering model is proposed in (Tillmann and Ney, 2003) which is a linear reordering model that simply skips a number of source words/phrases to allow the movement of the target translation with a particular penalty. However, this simple orientation model does not depend on the actual phrase itself but on the relative position between reordered phrases. More recently, a number of sophisticated reordering approaches have been proposed by (Tillmann, 2004), (Kumar and Byrne, 2005) and (Al-Onaizan and Papineni, 2006). These approaches focus on lexicalized reordering models which model the reordering based on the phrase itself not on the relative position as before. For example, the model can provide a probability for each phrase in a give context to be translated in monotone, swap with the neighbouring phrases or translate as discontinuous phrase and move further.

We think that the models presented above are satisfactory for modeling how to penalize the movement of the phrases; however, it depends on the language model to judge the grammaticality of the translation output with this movement. We think the n-gram language models limit the capability of reordering models since an n-gram language model cannot judge the grammaticality of a movement beyond the n-gram scope. We believe

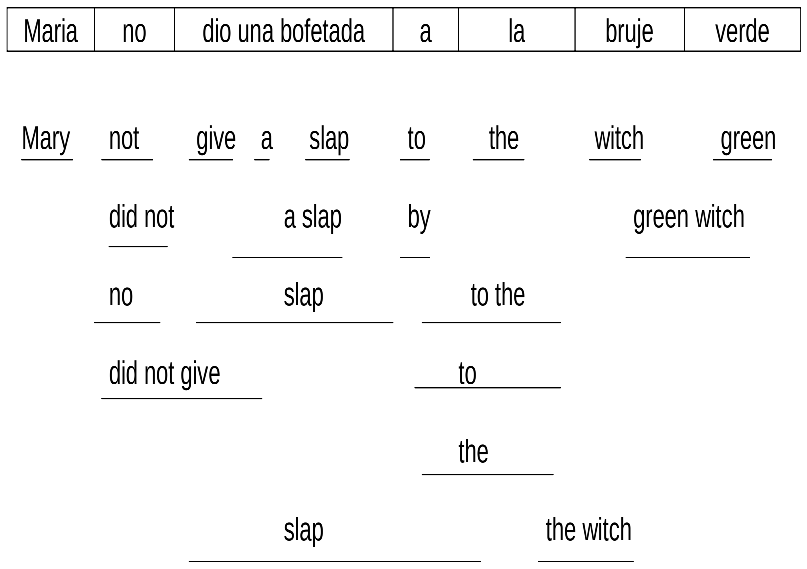


Figure 2.5: All possible source segmentations with all possible target translations (from (Koehn, 2004))

that more sophisticated language models can enable better reordering using the already proposed reordering techniques.

2.4.7 Phrase-based Decoder

The task of the Phrase-based SMT decoder is to search for the best translation given a source sentence, i.e. to maximize the probability as shown in the log-linear representation in Equation (2.4). Publicly available decoders like Pharaoh (Koehn, 2004a) and its open source successor Moses (Koehn et al., 2007) deploy a beam search decoder. The decoding starts by searching the phrase table for all possible translations for all possible fragments of the given source sentence. As shown in Figure 2.5, many possible segmentations for the source sentence along with many possible translations are available from the phrase table.

Starting with a null hypothesis, the decoder expands the hypothesis with the possible translations of the next source word (or phrase). The reordering is performed according to any of the approaches discussed in Section 2.4.6. Figure 2.6 shows possible expansions of the search space with translation candidates, where the cost of the translation path is accumulated together with pointer to the source covered words. The expansion process

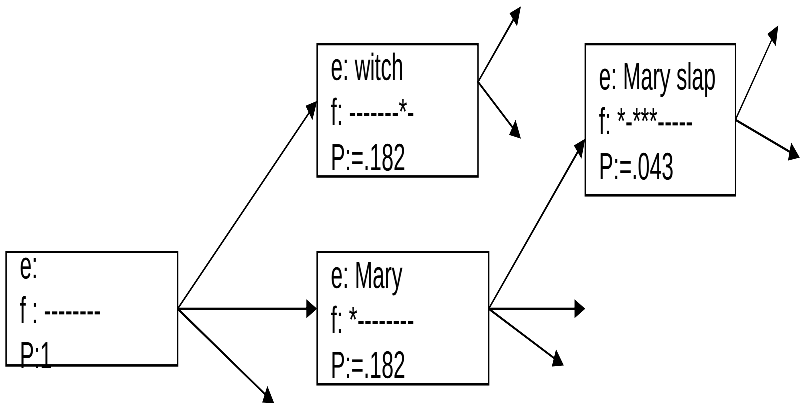


Figure 2.6: Expanding the decoder hypothesis with possible translations (from (Koehn, 2004))

continues untill there are no more uncovered source words. This large space has to be searched to obtain the best path.

The search space explodes exponentially due to the reordering and the large number of translation candidates; (Knight, 1999) showed that even decoding a word-based model with a bigram language model is an NP-Complete problem. Some strategies have to be used to limit the exponential explosion of the search space; therefore, a beam search pruning strategy is used to prune those hypotheses having a high cost and thus reduce the search space. Moreover, similar hypotheses are combined to reduce further the search space, if they cover the same source words and share the same language model history.

The decoder calculates a future cost estimation for the uncovered parts of the source sentence; at each hypothesis the future cost is estimated based on the translation cost and the language model cost of the uncovered source words. The total cost of the hypothesis is the sum of the actual cost and the future cost and thus the total cost can be a good estimation of the complete hypothesis cost. The decoder keeps a number of stacks to keep all partial translations of the target sentence, and the beam search pruning is applied to all such stacks to keep the most likely hypotheses. Finally the hypothesis that covers all source words with the lowest cost is chosen as the most likely translation.

2.5 Syntax support for Phrase-based SMT

There exist various approaches for enriching statistical models of translation with hierarchical structure, e.g. (Wu, 1997; Alshawi et al., 2000; Yamada and Knight, 2001; Koehn et al., 2003; Och et al., 2004; Chiang, 2005; Quirk et al., 2005; Marcu et al., 2006; Galley et al., 2006; Zollmann and Venugopal, 2006). We concentrate here specifically on related approaches that extend Phrase-based SMT systems by incorporating syntactic/hierarchical structure.

In contrast to (Koehn et al., 2003), who demonstrated that using syntax to constrain their phrase-based system actually harmed its quality, a number of researchers have, to different degrees, reported improvements when grammatical information is incorporated into their models of translation. We will review these approaches here.

2.5.1 Syntax Support via Reranking

The work described in (Och et al., 2004) is a significant attempt at including a variety of syntactic descriptions in a Phrase-based SMT system, including source language POS tags for improved reordering, parse tree probability, and tree-to-string, tree-to-tree, subtree-to-string and supertag-to-supertag features. Of these syntactic features, only the subtree-to-string and supertag-to-supertag features gave a modest improvement over the baseline system when they were included as features for reranking the n-best output of the baseline system.

2.5.2 Hierarchical Phrase-Based Translation

(Chiang, 2005) introduced an approach for incorporating syntax into Phrase-based SMT, targeting mainly phrase reordering. (Chiang, 2005) was the first work to demonstrate any improvement when adding hierarchical structure to Phrase-based SMT. In this approach, hierarchical phrase transduction probabilities are used to handle a range of reordering phenomena in the correct fashion. (Chiang, 2005) proposed a generalized form of the phrases where a synchronous context-free grammar is used to provide the ability of inserting a sub-phrase into a larger phrase. The derived transduction grammar does not rely on any linguistic annotations or assumptions, so that the 'syntax' induced is not linguistically motivated and does not necessarily capture grammatical preferences in the output target sentences. In fact all the phrases have a single generalization category and thus each phrase can be substituted for any other phrase and an *n*-gram language model is used to judge the resulting phrases. This approach requires a chart-based decoding which has much more computational cost than the beam search decoding used for Phrase-based SMT (cf. Section 2.4.7). Furthermore, (Chiang, 2005) used a small language model to avoid the complex search requirements when adding a large *n*-gram language model.

2.5.3 Syntactified Phrase-based MT (SPMT)

Even more recently, (Galley et al., 2006) and (Marcu et al., 2006) present two similar extensions of Phrase-based SMT systems with syntactic structure on the target language side. Both employ tree-to-string (so-called xRS) transducers, but their methods of acquiring the xRS rules and training them are somewhat different. In (Galley et al., 2006), the target subtrees are obtained by cutting up the syntactic trees into subtrees while maintaining a translation correspondence with the source language string. In (Marcu et al., 2006), 'syntactified' target language phrases are extracted by a traversal of the parse tree guided by manually specified rules regarding the likelihood of xRS target structure boundaries. Because of the conceptual and technical similarities between these two approaches, we next concentrate on the approach presented in (Marcu et al., 2006).

In (Marcu et al., 2006), it is demonstrated that 'syntactified' target language phrases can improve translation quality for Chinese–English. A stochastic, top-down transduction process is employed that assigns a joint probability to a source sentence and each of its alternative syntactified translations; this is done by specifying a rewriting process of the target parse-tree into a source sentence. The rewriting/transduction process is driven by xRS rules, each consisting of a pair of a source phrase and a (partially) lexicalized target sub-tree, termed a syntactified target phrase by (Marcu et al., 2006). This approach depends on inducing millions of xRS rules from parallel data,; however, they note that 28% of the phrase pairs cannot be directly associated with xRs rules, so that this large proportion of the phrase pairs can only be dealt with in an *ad hoc* manner. Similar to (Chiang, 2005), SPMT requires a chart-based decoding which has a high computational cost.

2.5.4 Syntax-Augmented Machine Translation

(Zollmann and Venugopal, 2006) extended the work introduced in (Chiang, 2005) by augmenting the hierarchical phrases with syntactic categories derived from parsing the target side of a parallel corpus. They associate a target parse tree for each training sentence pair with a search lattice constructed from the existing phrase translations on the corresponding source sentence. (Zollmann and Venugopal, 2006) used a parser to parse the target side of the parallel corpus to produce a syntactically motivated bilingual synchronous grammar like (Chiang, 2005). Similar to (Marcu et al., 2006), constituent target phrases are assigned the associated subtrees while heuristics are used to assign partial rewriting rules for the non-constituent phrases. Similar to (Chiang, 2005), a chart-based parser with a limited language model is used.

2.6 Direct Translation Models

2.6.1 Limitations of Log-Linear Phrase-based Model

First, we will define generative and discriminative modeling as two machine learning techniques. Both models use some input to produce some output, i.e. we want to learn a function to map $X- > Y$ which is equivalent to learning $P(Y|X)$.

A generative model is a probabilistic model that estimates a distribution over all inputs and outputs; this probability distribution is defined as a joint probability between all input and output variables. We then model $P(X|Y)$ and $P(Y)$; through the use of Bayes' rule we can estimate $P(Y|X)$. The generative models are so called because the distribution $P(X|Y)$ describes how to generate an input X for a given output Y. Generative models enjoy very computationally efficient methods for estimating the model parameters as they use maximum likelihood estimation directly on the observed data; however, the joint distribution limits the model capability as each input and output has to be modelled jointly.

On the other hand, the discriminative models provide a model of the output variables conditioned on the observed variables, i.e. they directly model $P(Y|X)$. These models are called discriminative since they can discriminate between different possible outputs given a particular input. This is usually done by defining a large number of feature functions on the input-output variables. The main disadvantage of the discriminative models is the high computational cost required for training the large space of parameters associated with the feature functions.

In the light of these definitions, the noisy-channel model is clearly a generative model where we model a joint probability of source and target words, and using Bayes' rule we estimate the translation probability. On the other hand, the log-linear representation of Phrase-based SMT is neither a generative model nor a discriminative model. In fact, it deploys a discriminative framework in which a limited number of features can be combined while using generative components as the feature functions. As we discussed in Sections 2.4.3 and 2.4.4, simplification assumptions were made to facilitate the parameter estimation process which led to limit the potential of the model.

The log-linear representation with MERT estimation has been widely used in Phrase-based SMT research since it has been introduced. However, two major drawbacks limit its utilization in modeling better MT systems. First, the parameters of the systems' components are independent and cannot be correlated. Second, the incapability of handling a large number of features; as a matter of fact, along with other researchers (Chiang et al., 2008), we think that the log-linear representation with the limited capability of MERT represents the bottleneck of further serious development of features rich SMT systems.

Fortunately, the log-linear representation of Phrase-based SMT was based on a fully discriminative SMT approach proposed in (Papineni et al., 1997, 1998). Next, we will review this approach along with a more recent development based on it (Ittycheriah and Roukos, 2007).

2.6.2 Direct Translation Model

(Papineni et al., 1997) proposed a Direct Translation Model (DTM) which models the *a posteriori* conditional distribution $P(T|S)$ as a discriminative model. DTM has three components: a prior conditional distribution $P_0(T|S)$, features that capture the translation and language model effects in a unified framework, and finally weights of the features that can be estimated by Maximum Entropy (Berger et al., 1996).

DTM provides a very powerful framework for modeling MT by utilizing a large number of features which can capture different levels of correlations between various effects in the MT system. Moreover, the estimation of the feature weights is fully data-driven. This representation turns the problem of MT into a sequential classification problem in which the classifier deploys various features from the source and candidate target translation to specify a sequence of decisions that finally result in an output target string.

As shown in Eqn (2.6), the feature functions $\phi_i(S,T)$ are defined over source and target. These feature functions may represent any view of the source and target phrases such as POS tags, parsing information and morphological information. Each feature function is associated with a weight λ_i which specifies how much this feature contributes to the overall translation probability. It is worth noting that the term Z in the formula is the normalization factor which is needed to produce a well-formed probability distribution. This term is responsible for the high computational cost of training Maximum Entropy models. Fortunately, the normalization factor is not required at decoding time as it is constant for a given S.

$$T^* = \arg\max_T P(T|S) = 1/Z \quad exp \sum_i \lambda_i \phi_i(S,T) \tag{2.6}$$

2.6.3 Direct Translation Model 2 (DTM2)

DTM2 Overview

Recently, (Ittycheriah and Roukos, 2007) introduced Direct Translation Model 2 (DTM2) which outperforms state of-the-art Phrase-based SMT systems by handling the Phrase-based SMT problem as a direct translation model, using minimum number of phrases with no overlap and finally training the whole set of millions of system parameters using the Maximum Entropy framework.

Direct Translation Model (DTM) models the *a posteriori* conditional distribution $P(T|S)$ instead of $P(S|T)$ as in the source channel approach. DTM has three components: a prior conditional distribution $P_0(T|S)$, features that capture the translation and language model effects in a unified framework and finally weights of the features that can be estimated by the Maximum Entropy (Berger et al., 1996) technique.

As shown in Equation (2.6), Phrase-based SMT is represented as a classification problem with arbitrary features defined over source and target. More specifically, the reordering and prior phrase probabilities are represented as shown in Equation (2.7).

$$P(T|S) = P_0(T,J|S)/Z \quad exp\sum_i \lambda_i \phi_i(T,J,S) \tag{2.7}$$

Here P_0 is the prior distribution for the phrase probability which is usually the phrase normalized counts used in any conventional Phrase-based SMT system, and J is the skip reordering factor for this phrase pair which represents the jump from the previous source word.

DTM2 Phrase Structure

Phrase extraction as outlined in Section 2.4.5 results in a huge phrase table with large overlaps between the extracted phrases, such that longer phrases overlap with smaller sub-phrases. (Chiang, 2005) extended the phrase-pairs to hierarchical phrase-pairs (cf.

Section 2.5), where a grammar with a single non-terminal allows the embedding of phrase pairs; however, the phrase pairs still have the overlap problem. DTM2 proposed a similar phrase structure to the one proposed in (Chiang, 2005) while solving the overlapping problem by maintaining the minimum possible number of phrases by following the concept introduced earlier in (Brown et al., 1993). Simply, a multi-word target phrase should be included if it is sufficiently different from a word-by-word translation. Figure 2.7 shows some examples of the phrase pairs in DTM2.

اللجنة	of the X committee
المركزية	central
للحزب	of the X Party

Figure 2.7: Phrase structures in DTM2. X represents a variable in the target phrase. From (Ittycheriah and Roukos, 2007)

DTM2 Features

DTM2 provides a flexible framework for any feature type. Currently it deploys five types of features:

- Lexical Features: these are micro features that examine source and target words of the phrases.
- Lexical Context Features: these features encode the context of the source and target phrases (i.e. previous and next source and previous target).
- Source Morphological Features: these features encode morphological and segmentation characteristics of the source words.
- Part-of-Speech Features: these features encode source and target POS tags effects as well as POS tags of the surrounding contexts.

DTM2 Decoder

The decoder adopted in (Ittycheriah and Roukos, 2007) is a beam search decoder similar to decoders used in standard Phrase-based log-linear systems such as (Tillmann and Ney, 2003) and (Koehn, 2004a). There are two main differences between DTM2 decoder and standard Phrase-based SMT decoders. First, DTM2 deploys Maximum Entropy probabilistic models to obtain the translation costs and various features costs by deploying the features described above. Second, the DTM2 decoder handles phrases with variables, as shown in Figure 2.7. When a decoding path is expanded with a phrase with variables, the next extensions of this path can either substitute this variable or further extend it with another phrase. The languge model can be added as a Maximum Entropy feature; however, this would limit the language modeling to the target side of the parallel corpus. To overcome this limitation, the translation model is combined with an *n*-gram language model as a log-linear combination to allow the use of language models built from a very large monolingual corpus.

2.7 MT Evaluation Overview

Evaluation of MT output is a very hard task as it is a subjective evaluation and there are no known measures that can easily be checked to indicate how good the translation is. Evaluation metrics have been proposed which try to measure the translation output while correlating with human judgments. Bilingual Evaluation Understudy (BLEU) which has been proposed in (Papineni et al., 2002) is the most widely used evaluation metric. BLEU score measures the translation quality by calculating the geometric means of *n*-gram agreements between the output translation and one or more reference translations. To account for the deletion of words and penalize translations with high precision but low recall, the BLEU score includes a brevity penalty factor that penalizes translations shorter than the references.

Many variations have been proposed to extend the BLEU score. For example, in

METEOR (Banerjee and Lavie, 2005) the focus is on recall by incorporating the use of stemming and synonyms from Wordnet to match similar target variations. More recently, an extension of BLEU score to measure the dependency relations between the translation and the references was proposed in (Owczarzak et al., 2007). Anyway, the automatic evaluation of MT remains a highly controversial issue due to the lack of an acceptable measure that can capture translation variations.

More recently, human evaluation such Human Translation Error Rate (HTER) is being used in large-scale evaluations. HTER is a human-based version of the Translation Error Rate (TER) metric, where a human calculates the minimum number of insertions, deletions and substitutions needed to correct the translation output according to some guidelines. While it is gaining acceptance, it is not available for everyday tasks for all researchers.

2.8 Summary

SMT has evolved from the word-based models (Brown et al., 1988, 1990) to Phrase-based models (Zens et al., 2002; Tillmann and Xia, 2003; Koehn et al., 2003). Then, motivated by seeking more grammatical translations and better reordering, researchers started to integrate syntax into Phrase-based SMT (Chiang, 2005; Marcu et al., 2006; Zollmann and Venugopal, 2006). More recently, DTM2 (Ittycheriah and Roukos, 2007) was proposed to allow for the integration of richer features into phrase-based SMT.

All the approaches proposed for incorporating syntax into Phrase-based SMT (Chiang, 2005; Marcu et al., 2006; Zollmann and Venugopal, 2006) share common drawbacks. First: they all use synchronous PCFG which does not match non-constituent phrases commonly used in Phrase-based SMT and therefore the approaches usually resort to some heuristics to annotate such phrases with syntactic structures. Secondly, all of them deploy chart-based decoders with a very high computational cost compared with Phrase-based beam search decoders. Third, the proposed approaches deploy small language models compared to what is usually used in Phrase-based systems to limit the decoding complexity.

In this thesis, we explore the possibility of improving Phrase-based SMT systems translation quality by incorporating syntactic structures on the target sentences while dealing with the non-constituent phrases. Furthermore, we will explore the possibility of incorporating syntactic structures into Phrase-based SMT systems without sacrificing the computational efficiency of the linear decoders or the capabilities of high-order language models. We will explore various levels of syntactic integration. First, incorporating lexical syntax translation model and *n*-gram language model into Phrase-based SMT is explored in Chapter 4. Second, in Chapter 5, we introduce a novel incremental dependency-based language model. Third, we incorporate the incremental dependency-based language model into SMT in Chapter 6. In the next chapter, we will introduce an over view of the lexical syntax and lexicalized grammars approaches used in this thesis.

Chapter 3

Lexical Syntax Overview

3.1 Syntax and Parsing

The syntax of a language defines the rules and principles that govern the grammatical structure of a sentence in that language. Syntax can define the grammaticality of a sentence on many different levels: constituency, dependency relations and logical/semantic structure.

Constituency is where a group of words function as a single unit within a structure, e.g. a noun phrase such as "the time of the elections" represents a constituent which acts as single unit and thus the syntax can describe the non-local reordering of constituents within a structure.

Dependency grammars model the relations between words or constituents in the syntax structure using the subcategorization information. For example subcategorization information of a di-transitive verb like "give" in a sentence like "I give him a pen" should encode that the verb has a subject "I" and two objects "him" and "a pen". Thus, the dependency grammars define the syntax structure as a set of grammatical relations between a word and its dependents.

Finally the syntax or the dependency structure can provide a logical representation between words by predicate argument relations as a semantic representation.

Having defined the role of syntactic structure, this is realized using parsing i.e. ana-

$$S \longrightarrow NP \quad VP \qquad : non - terminal$$

$$VP \longrightarrow Verb \quad NP \qquad : non - terminal$$

$$Verb \longrightarrow eat \qquad : terminal$$

Figure 3.1: CFG production rules.

lyzing a sentence to determine its grammatical structure. Context-Free Grammar (CFG) is the most commonly used syntactic representation for parsing. CFG is represented by production rules on terminals (words) and non-terminals that represent more generalized structures. Figure 3.1 shows some examples of CFG production rules. Probabilistic CFG (PCFG) assigns a probability for each of the grammar production rules. The state-of-the-art parsers (Collins, 1999; Charniak, 2000) are based on head-lexicalized PCFG.

3.2 Lexicalized Grammars

Modern linguistic theory proposes lexicalized grammars in which a syntactic parser has access to an extensive lexicon of word-structure pairs and a small set of operations to manipulate and combine the lexical entries into parses. The structures of a lexicalized grammar can be elementary trees, sub-graphs and etc. Each structure is associated with a lexical item, thus the whole grammar is defined on the lexicon which associates the lexical items to its corresponding structures. A finite set of operations is used to combine the elementary structures together. In contrast to other grammars such as CFG, there are no grammar rules defined on the non-lexical level at all.

Lexicalized grammars encode the dependency, syntactic and subcategorization information on the lexical level, such that the grammar localizes the long-range dependencies. Lexicalized grammars perform this localization by encoding all the arguments needed by the associated lexical item but no more arguments than that. Moreover, lexicalized grammars factor out recursive structures into different elementary objects. Consider example (3.1), where the elementary structure associated with the verb *resigned* should contain an argument which is linked to the subject *officer*. While this is a long-range dependency, it has been localized on the lexical level using the elementary structure.

The **officer** who is in charge of the operation **resigned**. (3.1)

These characteristics turned out to be pivotal for our approach of integrating syntax into SMT systems as we will discuss in this thesis. In this chapter, we review two lexicalized grammars: LTAG and CCG. Both have been used in the work done for this thesis.

3.3 Lexicalized Tree Adjoining Grammar

In Lexicalized Tree Adjoining Grammar (LTAG) (Joshi and Schabes, 1991) a lexical description is an elementary tree structure as shown in Figure 3.2. Each elementary tree represents a possible tree structure for the word. The elementary tree represents a complex syntax description that localized the syntactic and semantic (predicate-argument) constrains using subcategorization information. There are two kinds of elementary trees: initial trees and auxiliary trees. Initial trees are phrase structure trees which contain no recursion, while auxiliary trees represent phrase structure with recursion. Figure 3.2 shows initial trees denoted by α and auxiliary trees denoted by β. The LTAG subcategorization information is most clearly available in the verb *includes* which takes a subject NP to its left and an object NP to its right.

LTAG elementary trees can be combined using two operations, substitution and adjunction. The substitution operation is used to insert an initial tree into an elementary tree. The adjunction operation is used to attach an auxiliary tree to an elementary tree. In the lower part of Figure 3.2 the parse tree derived from combining the elementary trees by substitution and adjunction operations is shown. This parse tree is called the derived

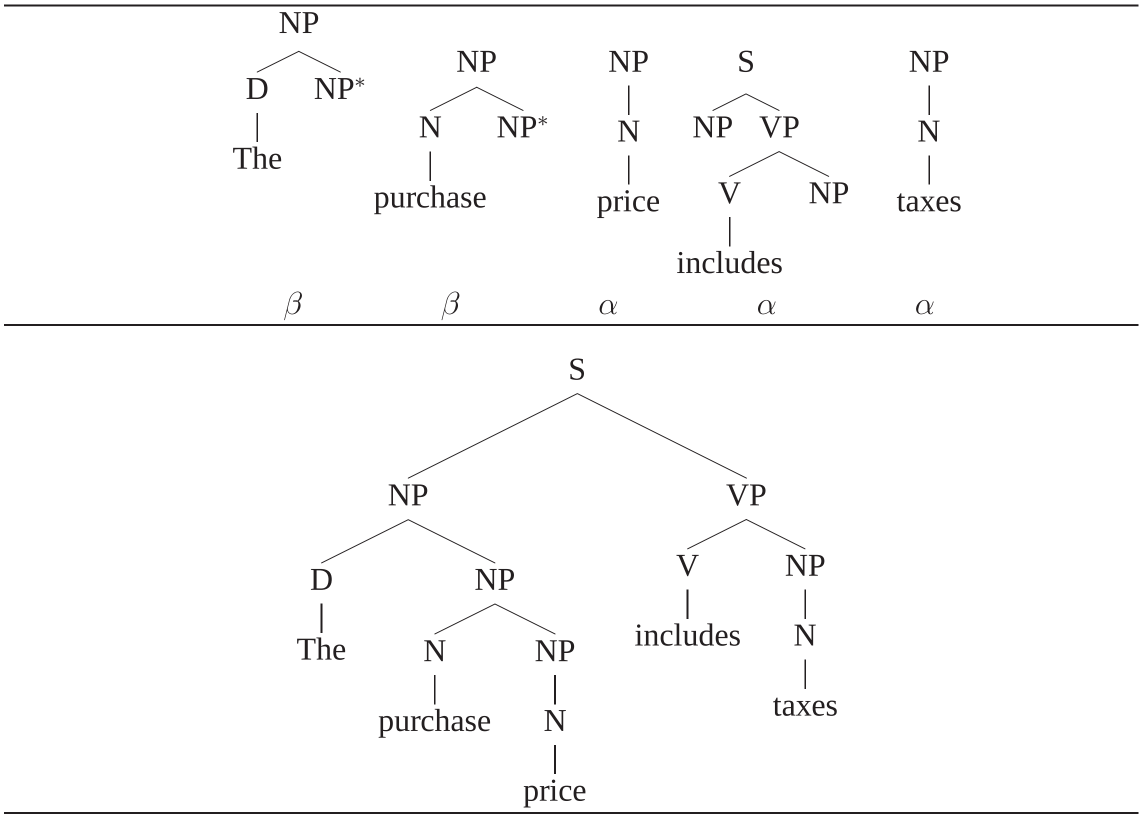


Figure 3.2: LTAG elementary trees for the sentence with the parse tree resulting from combining the elementary trees (from (Hassan et al., 2008a))

tree which represents the resulting phrase structure from combining the elementary trees during the derivation process. The derived tree neither encodes the elementary structures used in the derivation nor the operations used to combine them. On the other hand, the derivation tree encodes the elementary structures and the operations used during the derivation process. Figure 3.3 shows the derivation tree for the phrase structure shown in Figure 3.2.

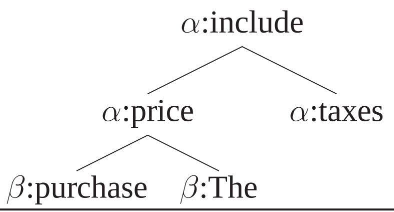


Figure 3.3: LTAG derivation tree that produced the parse tree in Figure 3.2. Initial trees are inserted using substitution and auxiliary trees are inserted by adjunctions.

3.3.1 LTAG Supertagging

The term "supertagging" (Bangalore and Joshi, 1999) refers to tagging each of the words of a sentence with a supertag which represents the elementary tree associated with the lexical item. When well-formed, an ordered sequence of supertags can be viewed as a compact representation of a small set of constituents/parses that can be obtained by assembling the supertags together using the appropriate combinatory operators (such as substitution and adjunction in LTAG). Similar to POS tagging, the process of *supertagging* an input utterance proceeds with statistics that are based on the probability of a word-supertag pair given their Markovian or local context (Bangalore and Joshi, 1999). When supertagging is performed, most of the ambiguity in constructing the parse structure is almost eliminated and the combinatory operators can be used to construct the structure using the assigned supertags (hence why this approach is called *'almost parsing'*). In fact, (Nasr and Rambow, 2004) have quantified the *'almost parsing'* to be 97.7% dependency accuracy of the full parsing accuracy when using the correct supertags. The main difference with full parsing is that supertagging the input utterance need not result in a fully connected graph.

An LTAG parser may be used to perform the adjunction and substitution operations to construct the syntactic tree. More efficiently, (Bangalore, 2000) proposed a simpler method to construct the tree called ”Light Weight Dependency Analyzer“ that can construct the derivation tree using the dependency information encoded in the supertags via deterministic methods.

The original LTAG-based supertagger of (Bangalore and Joshi, 1999) is a standard HMM tagger. A more recent version of the LTAG supertagger (Bangalore et al., 2005) conditions a supertag on a vector of features representing its context and employs a Maximum Entropy classifier (Berger et al., 1996). The supertags were extracted from the Penn Treebank (Marcus et al., 1993) by (Chen et al., 2006).

3.4 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) (Steedman, 2000) is a lexicalized grammatical theory based on Categorial Grammar and lambda-calculus (Ajdukiewicz, 1935; Bar-Hillel, 1953). The CCG lexical entries define syntactic categories which encode syntactic valency and directionality; these categories can be augmented by a semantic representation to provide compositional semantics with a completely transparent interface between surface syntax and logical semantics.

CCG syntactic categories can be associated with a semantic interpretation with the same type of the syntactic category. These semantic categories are represented in λ-calculus with predicate-argument information in a completely transparent interface with the syntactic representation such that it can provide compositional semantics through the syntactic structure. Although in this thesis we focus only on syntactic structures, CCG provides the possibility of expanding the work presented in this thesis to semantic representation as well.

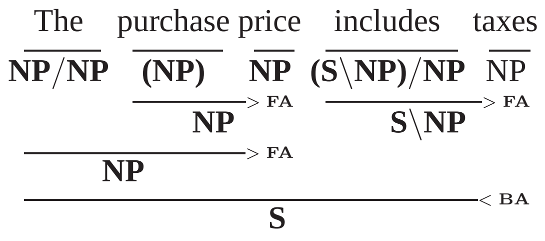


Figure 3.4: CCG Supertags and the derivation combining the supertags into a parse-tree.

Each word is associated with syntactic categories which define its syntactic behaviour in different contexts. There are two kinds of syntactic categories: atomic categories and complex categories. Atomic categories are simple categories such as S, NP, PP and N. Complex categories are functors; for example a complex category like X/Y will take argument Y to its right side resulting in category X.

As shown in Figure 3.4, the notation α/β and $\alpha\backslash\beta$ represents a predicate/functor α that expects an argument β to the right and left respectively. A sequence of supertags $[\beta \quad \alpha\backslash\beta]$ can be combined by Backward application resulting in α (similarly for Forward application $[\alpha/\beta \quad \beta]$). The derivation shown corresponds to the (upside-down) parse tree shown in the lower part of Fig. 3.2.

It is worth noting that any complete or partial derivations can by associated with structure which makes CCG very appealing for handling non-constituent phrase structure and for incremental parsing as well.

3.4.1 CCG Combinatory Operators

CCG has three types of operators: application operators, composition operators and type raising. We will review here some of the CCG operators.

Application Operators

- Forward Application (FA): this operator performs the forward application combinatory rule for CCG as defined in (Steedman, 2000). As shown in (3.2) and (3.3), if a constituent with category X/Y is immediately followed by a constituent with category Y, the operator FA can be used to combine them to construct a new constituent with category X.

$$\dfrac{\mathbf{X/Y} \quad \mathbf{Y}}{\mathbf{X}}{}^{>\text{FA}} \tag{3.2}$$

$$\begin{array}{cc} \text{The} & \text{cat} \\ \mathbf{NP/NP} & \mathbf{NP} \\ \hline \multicolumn{2}{c}{\mathbf{NP}} \end{array}{}^{>\text{FA}} \tag{3.3}$$

- Backward Application (BA): this operator performs the backward application combinatory rule, as defined for CCG (Steedman, 2000), on two categories. If a constituent with category $X\backslash Y$ is immediately preceded by a constituent with category Y, the operator BA can combine them to construct a new constituent with category X, as illustrated in examples (3.4) and (3.5).

$$\dfrac{\mathbf{Y} \quad \mathbf{X\backslash Y}}{\mathbf{X}}{}^{<\text{BA}} \tag{3.4}$$

$$\begin{array}{cc} \text{John} & \text{sleeps} \\ \mathbf{NP} & \mathbf{S\backslash NP} \\ \hline \multicolumn{2}{c}{\mathbf{S}} \end{array}{}^{<\text{BA}} \tag{3.5}$$

Compositional Operators

- Forward Composition(FC): this operator is the compositional version of the FA operator and it performs the Forward Composition combinatory rule on two complex categories, as defined in (Steedman, 2000). Consider (3.6) and (3.7), where if a constituent with category X/Y is immediately followed by a constituent with category Y/Z, the operator FC can combine them to construct a new constituent with category X/Z.

$$\begin{array}{cc}\mathbf{X/Y} & \mathbf{Y/Z}\\ \hline \multicolumn{2}{c}{\mathbf{X/Z}}\end{array}{}^{>\,\mathrm{FC}} \tag{3.6}$$

$$\begin{array}{ccc}\text{The} & \text{tall} & \text{man}\\ \mathbf{NP/NP} & \mathbf{NP/NP} & \mathbf{NP}\\ \cline{1-2} \multicolumn{2}{c}{\mathbf{NP/NP}} & \end{array}{}^{>\,\mathrm{FC}} \tag{3.7}$$

- Backward Composition (BC): this operator is the compositional version of the BA operator and it performs the Backward Composition combinatory rule as defined in (Steedman, 2000). If a constituent with category Y/Z is immediately followed by a constituent with category $X\backslash Y$, the operator BC can combine them to construct a new constituent with category X/Z, as illustrated in examples (3.8 and 3.9).

$$\begin{array}{cc}\mathbf{Y/Z} & \mathbf{X\backslash Y}\\ \hline \multicolumn{2}{c}{\mathbf{X/Z}}\end{array}{}^{<\,\mathrm{BC}} \tag{3.8}$$

$$\frac{\mathbf{NP/PP} \quad \mathbf{S\backslash NP}}{\mathbf{S/PP}}{<}\,\mathrm{BC} \tag{3.9}$$

Type Raising (TR):

Type raising captures long-range dependencies and is usually used with composition operators. (Steedman, 2000) defines Type Raising as a unary rule. If a constituent with category X/Y is immediately preceded by a constituent with category Z such that X/Y has a long-range dependency on the right side to a category Z, then Type Raising is used to raise the category Z to category Y.

Examples (3.10) and (3.11) demonstrate type raising with forward composition, the subject NP is type raised to S and then forward composed with $(S\backslash NP)/NP)$ to compose S/NP.

$$\frac{\mathbf{X} \quad \mathbf{(Y\backslash X)/Z}}{\mathbf{X/Z}}\,\mathrm{T} \tag{3.10}$$

$$\frac{\begin{array}{cc}\text{He} & \text{bought}\\ \mathbf{NP} & \mathbf{(S\backslash NP)\ /NP}\end{array}}{\mathbf{S\ /NP}}\,\mathrm{T} \tag{3.11}$$

3.4.2 CCG Supertagging

Based on the supertagging approach in (Bangalore and Joshi, 1999), (Clark and Curran, 2004) introduced a CCG supertagger using Maximum Entropy classification techniques. The CCG supertags are the atomic and complex categories associated with each word. (Clark and Curran, 2004) used supertagging before parsing to achieve accurate and efficient parsing results. Similar to the LTAG supertagging, the supertagger uses statistical sequence tagging techniques to assign a limited number of lexical categories to each word in the sentence and therefore the parser can search in much reduced space to assign the parse structure. The CCG supertags were automatically extracted from the Penn Treebank (Marcus et al., 1993) by (Hockenmaier and Steedman, 2007).

3.5 Comparison between LTAG and CCG

Many researchers have indicated the similarity between LTAG and CCG since both grammars are mildly context-sensitive grammars. In fact, the lexical descriptions of both grammars are equivalent, i.e. the LTAG elementary trees are equivalent to CCG categories such that the same dependency arguments are represented at both LTAG elementary trees and CCG categories as well. Based on this similarity, (Doran and Bangalore, 1994) introduced a methodology for bootstrapping CCG categories from LTAG elementary trees. However, as they pointed out the LTAG derived trees represent a more rigid structure than CCG flexible derivations. In the context of supertagging, this difference seems minor and not so relevant; while for full parsing these differences are more crucial. On the one hand, the more flexible CCG derivations complicate the parsing process as more structures should be considered (Clark and Curran, 2007). On the other hand, the more flexible derivations can facilitate incremental and partial parsing (Hockenmaier and Steedman, 2007).

The CCG Combinatory Operators assemble lexical entries together into derivation-trees; each partial or complete syntactic derivation corresponds directly to a structure. For example, strings such as *"John likes"* have a natural interpretation as constituents. (Doran and Bangalore, 1994) highlighted that the flexibility of CCG derivations allows the handling of non-constituent constructions that LTAG cannot handle, which is due to the fact that LTAG trees represent rigid structures while CCG categories allow more flexibility in the derivation process. Unlike many other linguistic theories, this flexibility gives CCG an advantage over other grammatical formalism in handling non-constituent constructions. It is worth noting that the capability of CCG to handle non-constituent constructions comes at a price, namely introducing more spurious parses and accordingly leading to more complicated parsing than LTAG.

3.6 Syntactic Language Models

Our main interest in lexicalized grammar is to develop syntactic translation and language models for enhancing MT output. In this section we will review the previous work of syntactic language models and then discuss the potential of lexical syntax for developing language models for machine translation.

3.6.1 Previous work

A number of researchers have introduced work that incorporates syntactic language models into speech recognition systems. The Structured Language Model (Chelba, 2000) and (Xu et al., 2002) proposed an incremental shift-reduce parser which conditions the probability of words on previous lexical heads, rather than previous words as in *n*-gram language models. The probability of the word is the weighted sum of its conditional probabilities from possible parses.

(Roark, 2001) proposed an incremental top-down and left-corner parser that generates conditional word probabilities. He deployed parse probabilities directly to calculate the string probabilities. (Collins et al., 2005) extended (Roark, 2001) by using a discriminative approach to estimate the model with more syntactic features.

(Charniak, 2001) proposed a head-driven parsing approach that directly used generative PCFG models as language models which made use of a non-incremental, head-driven statistical parser to produce string probabilities. (Charniak et al., 2003) integrated the model proposed in (Charniak, 2001) into a syntax-based MT system (Yamada and Knight, 2001) on a very small scale.

All the previous approaches depend on non-deterministic techniques to grow a huge number of partial derivations which is unmanageable for large-scale applications such as MT. This has limited the usability of these approaches to very small tasks and/or re-ranking of systems outputs. Another major aspect is that most of the previous approaches deploy PCFG which can not handle non-constituent constructions commonly used in Phrase-based SMT systems (cf. Section 2.4). Moreover, PCFG is not lexicalized and thus cannot naturally provide a complete account for lexical and syntactic effects. Lexicalized modifications of PCFG complicates the parsing process further.

3.6.2 Lexical Syntax Language Models

Lexical syntax offers a very appealing representation for language modeling since it has four distinct advantages that can help in providing efficient language modeling. First, all syntactic information is localized on the lexical level and thus it can match word or multi-word level. Second, since lexical syntax localizes the dependency information, there is no need to encode more complicated syntactic information on the higher level of the structures. This is a major advantage, since it can allow adding syntax without explicit need for high-level search of the possible structures (i.e. chart parsing). Third, lexical syntax can seamlessly provide dependency information using the subcategorization information encoded in the categories; therefore dependency information is represented on the lexical level. A language model can make use of such information to estimate the most likely word sequence based on satisfying the dependency relations. Fourth, as we discussed previously, supertagging can limit the ambiguity in the possible structures and therefore lexical syntax with supertagging will not grow a huge number of partial derivations when scoring possible structures for a language model, in contrast to the case with previously discussed syntactic language models.

3.7 Summary

Lexical syntax represents a very appealing grammatical formalism for exploring different forms of syntactic language models to enhance Phrase-based SMT systems. In this thesis, we will examine the utilization of lexical syntax in two forms. First, incorporating supertagging in the translation model and an *n*-gram supertagging language model into Phrase-based SMT is explored in Chapter 4. Second, in Chapter 5, we introduce a novel syntactic language model based on incremental CCG parsing. Third, we incorporate the incremental parsing language model into SMT in Chapter 6.

Chapter 4

Syntactic Phrase-based SMT: The Supertagging Approach

4.1 Introduction

Recently, SMT researchers have started to focus on extending Phrase-based Statistical Machine Translation (henceforth Phrase-based SMT) systems with syntactic knowledge; however, some of the early attempts caused system performance to deteriorate (Koehn et al., 2003). The most recent successful enrichment of Phrase-based SMT with hierarchical structure either employ non-linguistically motivated syntax for capturing hierarchical reordering phenomena (Chiang, 2005) or extend the phrase translation table with redundantly ambiguous syntactic structures over phrase pairs (Marcu et al., 2006; Zollmann and Venugopal, 2006).

In this thesis, we study the question as to whether the *lexical* descriptions developed in linguistic theory can benefit the translation quality of Phrase-based SMT system by improving the syntactic structure of the target sentences. We explore various levels of syntactic integration in a Phrase-based SMT system. First, incorporating supertags into the translation model and the language model of Phrase-based SMT is explored in this chapter. Second, in Chapter 5, we introduce a novel incremental dependency-based language model based on incremental CCG parsing. Third, we introduce incremental dependency-based SMT model by incorporating incremental parsing into the translation model along with incremental dependency-based language model, as described in Chapter 6.

In section 4.2 of this chapter, we discuss the main problems regarding integrating syntax into Phrase-based SMT and demonstrate how lexical syntax ia able to resolve these issues. In the rest of this chapter, we introduce our approach for integrating supertags into the translation model and the language model of Phrase-based SMT.

4.2 Syntax and Phrase based SMT

Over the last few years, Phrase-based SMT has been the most dominant paradigm within the field of Machine Translation (MT). As we discussed in Section 2.4, Phrase-based SMT systems demonstrate better accuracy and scalability than any other MT paradigm. However, it has proven difficult to incorporate linguistically motivated syntactic knowledge in order to obtain better quality translation output from Phrase-based SMT systems.

In Section 2.5, we reviewed various approaches for incorporating syntax into Phrase-based SMT. For example, (Koehn et al., 2003) demonstrated that adding syntactic constraints harmed the quality of their Phrase-based SMT system. (Och et al., 2004) explored the effectiveness of deploying a large set of syntactic features for re-ranking the translation output; only lexicalized subtrees and supertags gave a modest improvement among all features. (Chiang, 2005) induced hierarchical rules over the phrases that could capture a number of reordering phenomena. However, the induced syntactic structures are not linguistically motivated and do not necessarily capture grammatical preferences. More recently, (Marcu et al., 2006) employed a constrained yet syntactically justified phrase-translation table in which the target language side of a phrase pair constitutes a partially lexicalized syntactic structure. They induced millions of syntactic structures associated with the target phrase table; however, they resorted to some heuristics to obtain syntactic structures for the non-constituent target phrases which represent a huge part (28%) of their phrase table. Finally, (Zollmann and Venugopal, 2006) extended the work introduced in (Chiang, 2005) by augmenting the hierarchical phrases with syntactic categories derived from parsing the target side of a parallel corpus. Similar to (Marcu et al., 2006), heuristics were used to assign partial rewriting rules to the non-constituent phrases.

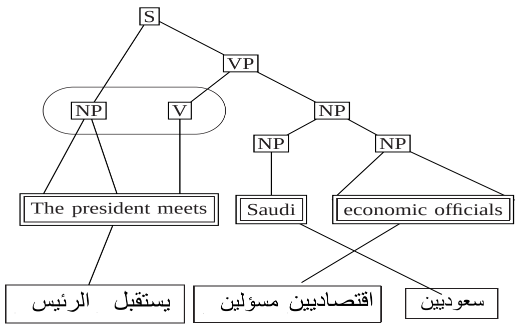


Figure 4.1: Arabic and English aligned phrase pairs with the English constituent structure (from (Hassan et al., 2008a)).

One major difficulty in extending Phrase-based SMT systems with linguistically motivated syntax is the mismatch between the notion of a ‘phrase’ in Phrase-based SMT systems (any sequence of words defined by the word alignment mapping) and the notion of a syntactic constituent in traditional linguistics. We present an example in Figure 4.1, which demonstrates clearly that while the first Arabic–English chunk alignment contains both the English subject NP as well as the main verb *“The president meets”*, this is not conventionally accepted as a constituent in English syntax. In contrast, in the same example, we see that the part of the object NP *“economic officials”* —the 3rd English chunk which maps to the second Arabic chunk— is usually interpreted as a constituent in English syntax. The problem is that the phrases in Phrase-based SMT systems are identified with regard to word alignment probabilities which need not follow any linguistic convention.

Figure 4.2 exemplifies the problem of associating a tree structure with non-constituent phrases such as those commonly assumed in treebanks and existing parsers. If we tried to associate a subtree with the non-constituent phrase *“The president meets”*, the subtree must also include the three encircled nodes in the figure. These three nodes constitutes an object *NP* (marked obj) and two adjuncts (a *PP* and the other encircled *NP*). Generally

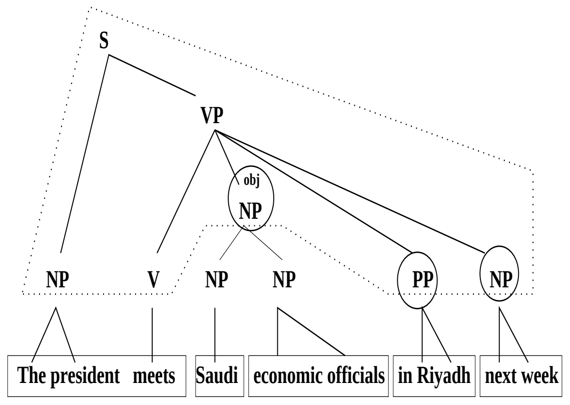


Figure 4.2: Mismatch between PBSMT phrases and tree structures (from (Hassan et al., 2008a)).

speaking, any of the sentences accepted by the regular expression `(The president meets NP [PP]* [NP]*)` that occurs in the training data will imply a new subtree for the phrase in question. The resulting subtrees differ only with respect to the number of adjuncts included under the *VP*. We refer to this as redundant syntactic ambiguity, because these subtrees unnecessarily partition the contexts and statistics of this phrase. Such redundancy represents an obstacle for any generalization to new instances and complicates the model both statistically and computationally. It is worth noting that this redundant subtrees are not lexicalized; usually these subtrees would be lexicalized which could lead to even more redundancy.

Due to such mismatches, it is not directly clear how the SMT notion of a phrase may be extended with a *tree* structure without introducing redundant ambiguity. For a non-constituent phrase, a tree structure representation directly introduces redundant ambiguity; multiple, alternative subtrees will be associated with the same phrase, whereas they are merely minor variants of each other, differing only in subgraphs that denote very specific contexts of the phrase.

The alternative, therefore, is to look for syntactic descriptions that do not produce redundant ambiguity in the phrase translation pairs. These syntactic descriptions must ensure an adequate and efficient representation of syntactic constraints on the word/lexeme level; moreover, these syntactic descriptions should be able to localize the global dependencies on the local, word/lexeme level. Accordingly, we explore a syntactic localization of Phrase-based SMT systems based on lexicon-driven approaches to linguistic syntax, i.e. Lexicalized Tree-Adjoining Grammar (Joshi and Schabes, 1991) and Combinatory Categorial Grammar (Steedman, 2000). In these linguistic approaches, it is assumed that the grammar consists of a very rich lexicon and a small set of combinatory operators that assemble lexical entries together into parse-trees. These operators neither carry nor presuppose further linguistic knowledge beyond what the lexicon contains. The lexical entries consist of syntactic constructs (supertags) that describe such lexical information as the POS tag of the word, its subcategorization information and the hierarchy of phrase categories that the word projects upwards in the parse-tree.

In this chapter, we present a syntactic lexicalization of Phrase-based SMT systems based on supertags called Supertagged Phrase-based SMT, which provides *'almost parsing'* to Phrase-based SMT by incorporating supertags in the translation model as well as in the *n*-gram language model.

4.3 Supertagging for Phrase-based SMT

Lexical syntax deploys rich syntax descriptions —supertags— that match individual words, and a limited set of Combinatory Operators which are used to combine supertags into a set of constituents/derivations. The supertagging language model (Bangalore and Joshi, 1999) replaces the set of combinatory operators with the more robust and efficient, statistical *n*-gram language model over the sequence of supertags (thus *'almost parsing'*). Supertagging language models can be implemented using finite-state technology, e.g. Markov Models, using probabilities based on the local context of the supertags such that an approximation to the syntactic structure is provided. There are currently two supertagging approaches: LTAG-based (Bangalore and Joshi, 1999) and CCG-based (Clark and Curran, 2004); the reader is referred to Chapter 3 for a thorough account of supertagging, LTAG and CCG.

Supertagging has two very interesting properties which make it especially suitable for extending Phrase-based SMT with syntax. Firstly, a supertag sequence can be constructed for any phrase found in a text, whether the phrase corresponds to a syntactic constituent or not. This implies that the target side of the phrase pairs can be augmented by supertags in a straightforward manner by annotating the parallel corpus with supertag sequences. Secondly, a supertag provides an extended *lexical* description of the neighbourhood and dependents of a word. Therefore, co-occurrence statistics over supertags may provide a good approximation of the syntactic validity of a concatenation of two phrases, leading to more fluent output. In addition to integrating the Markovian supertagging approach in Phrase-based SMT, we explore the utility of a new surface grammaticality measure based on combinatory operators.

In this chapter, we examine the effectiveness of extending a state-of-the-art Phrase-based SMT system with both LTAG and CCG supertags on the target language side. In order to avoid sparseness, we smooth the supertagged components (both language model and target side of the translation table) with backed-off components, including the components of a standard Phrase-based SMT system.

The remainder of the chapter is organized as follows. In section 4.4 we discuss the differences between Supertagged Phrase-based SMT and previous work on enriching Phrase-based SMT systems with syntactic structure. In section 4.5, we detail our approach. Section 4.6 describes the experiments carried out, together with the results obtained. Section 4.7 concludes, and discusses open questions.

4.4 Why Supertagged Phrase-based SMT?

There exist various approaches to incorporate syntax into Phrase-based SMT, e.g. (Chiang, 2005), (Marcu et al., 2006) and (Zollmann and Venugopal, 2006). As we have reviewed these approaches in Section 2.5, we will focus here on comparing our approach with (Marcu et al., 2006) and (Zollmann and Venugopal, 2006) which has extended (Chiang, 2005) by adding syntactic categories derived from parsing the target side of the parallel data.

While the work described in (Marcu et al., 2006) and (Zollmann and Venugopal, 2006) have much in common with the approach proposed in this chapter (such as the syntactified target phrases), there remain a number of significant differences. Firstly, both approaches deployed *ad hoc* manually specified rules to induce parse-chunks to accompany phrases; however, in our approach we utilize a more sophisticated and formalized syntactic representation that localizes global syntactic information and deploys rich lexical descriptions that match individual words/lexemes straightforwardly. A supertag can, therefore, be assigned to every word in a phrase. On the one hand, the correct sequence of supertags could be assembled (using only a small set of combinatory operators) into a small set of constituents/parses (*'almost parsing'*). On the other hand, because supertags are lexical entries, they facilitate robust syntactic processing (using Markov models, for instance) which does not necessarily aim at building a fully connected graph and which avoids redundant structural ambiguity.

A second major difference with (Marcu et al., 2006) and (Zollmann and Venugopal, 2006) is that our supertag-enriched source–target phrase pairs have not been generalized into any transduction rules that work with abstract categories. Such transduction rules are usually aimed at providing a treatment of phrase reordering. While it is certainly possible to extend our approach towards a transduction system, our model is currently targeted at more grammatical output given the standard reordering techniques used in mainstream Phrase-based SMT systems (cf. 2.4.6).

Thirdly, our model works with fully lexicalized syntactic descriptions and retains all phrase pairs used by the standard Phrase-based SMT system enriched with linguistic syntactic descriptions. Fourthly, supertagging is more efficient than actual parsing or tree transduction both in training and at run-time. Fifthly, we deploy a log-linear, left-to-right decoder (Tillmann and Ney, 2003), unlike (Marcu et al., 2006) and (Zollmann and Venugopal, 2006) who used a CKY-style decoder with high computational cost and small language models. Finally, unlike both approaches, we have no need to resort to *ad hoc* tree-rewriting measures in order to provide a better interaction between ('normal' Phrase-based SMT) and transduction rules.

In this chapter, we describe a different approach to obtaining more grammatical output using supertags, and provide stronger evidence for their effectiveness on large data sets for more language pairs. This chapter describes in detail our previous work in (Hassan et al., 2006), where we gave preliminary results on the effectiveness of our method using LTAG supertags, and (Hassan et al., 2007b), where we added work on CCG, used larger data sets, used a different decoder, showed greater improvements, as well as providing details on possible upper bounds with the method, and discussions on how the two supertaggers might be combined into one model; and finally, our work in (Hassan et al., 2008a) provided experiments on German–English translation and a more thorough analysis of the system.

Following our initial results on integrating supertags into Phrase-based SMT (Hassan et al., 2006), (Birch et al., 2007) presented a factored approach for Dutch–to–English employing CCG supertags as one of the factored translation models in a log-linear model in a completely different fashion than in the present work; they employed the supertags as a factored translation model as implemented in Moses (cf. (Koehn et al., 2007)). Positively, they report improved translation output when supertags are included on the target language side, and also when (separately) they are included on the source language side. Interestingly, however, in an analysis of the empirical results, (Birch et al., 2007) conclude that most of the improvement given by supertags can be obtained when using an improved reordering model. While we do not exclude the possibility that better reordering is one of the ways in which supertags improve over standard Phrase-based SMT systems, our empirical analysis in section 4.6 indicates that this accounts for only about 20% of the cases in which supertags provide improved output. In any case, we believe that supertags constitute a more promising, linguistically motivated method for improving reordering based on the existing reordering techniques as it can provide better language modeling for judging the movements proposed by the reordering models which cannot usually be judged by an *n*-gram language model.

4.5 Our Approach: Supertagged Phrase-based SMT

We extend the baseline Phrase-based SMT described in Section 2.4 with lexical syntactic representations (*supertags*) (cf. Sections 3.3 and 3.4) both in the language model as well as in the phrase translation model.

4.5.1 A Supertag-Based SMT model

Our baseline system is based on the system described in Section 2.4 where bidirectional word alignments are used to obtain lexical phrase translation pairs using heuristics presented in (Och and Ney, 2003) and (Koehn et al., 2003) (cf. Section 2.4.5 for more details).

Our extension of the baseline model includes supertags both in the phrase translation table and in the language model. As for the translation table, we employ an LTAG supertagger (cf. Section 3.3) and a CCG supertagger (cf. Section 3.4) to enrich the English side of the parallel training corpus with the 1-best supertag sequence per sentence. Then we extract phrase pairs together with the co-occurring English supertag sequence from this corpus using the usual phrase extraction method in Phrase-based SMT (cf. Section 2.4.5). For each extracted lexical phrase pair, we extract the corresponding supertagged phrase pairs from the supertagged target sequence in the training corpus. For each lexical phrase pair, there is at least one corresponding supertagged phrase pair, i.e. a phrase pair in which the target phrase is supertagged. It is worth noting that the word alignment is done on the lexical words only as in the baseline system. The target side of the training corpus is augmented with supertag sequences after the alignment process then the phrase extraction is performed on the supertagged target side of the training data to extract phrases with associated supertag sequences.

As for the supertagged language model, we employ the two aforementioned supertaggers to provide supertag sequences for a very large monolingual English corpus, from which we train a 5-gram language model over supertags to acquire an HMM supertagger (Bangalore and Joshi, 1999). This provides us with two HMM supertagging systems (CCG and LTAG) which are trained on large amount of monolingual target language data. This reduces the problems of sparseness in the lexical model, and provides useful language model probabilities for integration within our supertagged Phrase-based SMT model, described in the next section.

Using the supertagged translation table and language model, acquired as described above, we proceed with extending the baseline model described in Section 2.4. Next we define the probabilistic model that accompanies this syntactic lexicalization of the baseline model.

Let ST represents a supertag sequence of the same length as a target sentence t. Because the target sentences in the parallel corpus are now supertagged, we extract *supertagged phrase pairs*, i.e. phrase pairs in which the target phrase is supertagged. We will use the same notation σ for a segmentation into supertagged phrase pairs ϕ_s and $\phi_{t,ST}$ just as in the standard Phrase-based SMT (cf. Section 2.4).

In our model formulation, we employ the noisy-channel approach as the background against which we specify the log-linear formulation. The noisy-channel formulation would extend the noisy-channel model described in Section 2.4 as in Equation(4.1):

$$\begin{aligned} \arg\max_{t} \sum_{ST} P(s \mid t, ST) P_{ST}(t, ST) &\approx \\ \arg\max_{t,ST} P(s \mid t, ST) P_{ST}(t, ST) &\approx \\ \arg\max_{\sigma,t,ST} \overbrace{P(\phi_s \mid \phi_{t,ST})}^{TM\ w.sup.tags} \overbrace{P(O_s \mid O_t)^{\lambda_o}}^{distortion} \overbrace{P_{ST}(t, ST)}^{LM\ w.sup.tags} & \end{aligned} \tag{4.1}$$

In the first approximation we decide to avoid the complexity of summing over the supertag sequences for a target sentence. In the second approximation, just as in the baseline model, we do not sum over segmentations into phrases and their order (i.e. derivations), but rather again take a computationally more attractive approximation. These approximations carry over to the log-linear model formulation that is described next as an extension of equation the baseline log-linear formulation as in (4.2) which has been described in Section 2.4.

$$t^* = \arg\max_{t,\sigma} \prod_{f \in F} H_f(s,t,\sigma)^{\lambda_f} \tag{4.2}$$

Because we do not sum over supertag sequences in (4.1), the feature weight functions $H_f(s,t,\sigma,ST)$ in the log-linear model formulation (equation (4.2)) now have access to sequences of target language supertag sequences ST, as in (4.3):

$$t^* = \arg\max_{t,\sigma,ST} \prod_{f \in F} H_f(s,t,\sigma,ST)^{\lambda_f} \tag{4.3}$$

Our model interpolates (log-linearly) a novel set of *supertagged features* with the features of the baseline model. More formally, our model employs a feature set $F' = F \cup F_{st}$ that extends the standard Phrase-based model's feature set F (listed in Section 2.4) with the following set of supertagged features (F_{st}):

- [$_{lm.st}$] The function $H_{lm.st}(s,t,\sigma,ST) = P(ST)$ is a Markov supertagging model over sequences of supertags as in (4.4):

  $$P(ST) = \prod_{i=1}^{n} p(st_i | st_{i-4}^{i-1}) \tag{4.4}$$

  Here $ST = st_1 \dots st_n$. The parameters $p(st_i | st_{i-4}^{i-1})$ are estimated using Maximum-Likelihood with Kneser-Ney smoothing (Kneser and Ney, 1995). Note that because the five-grams in this model are over supertags, this model should suffer less from data sparseness than a five-gram language model over words. In what follows, we will refer to this Markov model over supertag sequences by the term 'supertagged language model'.

- [$\phi.st, r\phi.st$] A weight function $H_{\phi.st}(s,t,\sigma,ST) = P(\phi_s \mid \phi_{t,ST})$ and its reverse $H_{r_{\phi.st}}(s,t,\sigma,ST) = P(\phi_{t,ST} \mid \phi_s)$. The supertagged phrase translation probability is approximated in the usual way:

$$P(\phi_s \mid \phi_{t,ST}) \approx \prod_{\langle s_i, t_i ST_i \rangle \in (\phi_s \times \phi_{t,ST})} p(s_i \mid t_i, ST_i) \tag{4.5}$$

$$P(\phi_{t,ST} \mid \phi_s) \approx \prod_{\langle s_i, t_i ST_i \rangle \in (\phi_s \times \phi_{t,ST})} p(t_i, ST_i \mid s_i) \tag{4.6}$$

  In both (4.5) and (4.6), $\langle s_i, t_i, ST_i \rangle$ is a supertagged phrase pair consisting of the phrase pair $\langle s_i, t_i \rangle$ in which t_i is supertagged with ST_i. As usual, the parameters $p(s \mid t, ST)$ and $p(t, ST \mid s)$ are estimated by means of the relative frequency in the multiset of all supertagged phrase pairs extracted from the parallel corpus, as in (4.7) and (4.8) :

$$p(s \mid t, ST) = \frac{count(s,t,ST)}{\sum_s count(s,t,ST)} \tag{4.7}$$

$$p(t, ST \mid s) = \frac{count(s,t,ST)}{\sum_{t,ST} count(s,t,ST)} \tag{4.8}$$

- [Smoothing: $x.\phi.st, x.r\phi.st$] We employ two more feature functions ($x.\phi.st$ and $x.r\phi.st$) capturing the statistics $p(s_i \mid ST_i)$ and $p(ST_i \mid s_i)$, which in effect smooth the feature functions $\phi.st$ and $r\phi.st$. Because the baseline phrase-table probability ($p(s_i \mid t_i)$) is also a feature function in our model, interpolating with $p(s_i \mid ST_i)$ can be seen as smoothing $p(s_i \mid t_i, ST_i)$ using the approximation $p(s_i \mid t_i, ST_i) \approx p(s_i \mid t_i) \times p(s_i \mid ST_i)/p(s_i)$, where the probability of the source $p(s_i)$ is discarded as it does not alter the maximization over supertagged target sequences. Similarly, the feature $p(ST_i \mid s_i)$ can be seen to smooth the reverse probability $p(t_i, ST_i \mid s_i)$

in equation (4.6) as in the approximation $p(t_i, ST_i \mid s_i) \approx p(t_i \mid s_i) \times p(ST_i \mid s_i)$. A model in which we omit these two smoothing components (i.e. $p(s_i \mid ST_i)$ and $p(ST_i \mid s_i)$) turns out to be less optimal than this formulation (but still outperforms the baseline system).

It is important to highlight that the interpolation of the language and phrase-translation table component features of the baseline model (lm, ϕ and r_ϕ) in our model can be seen as smoothing of the corresponding supertagged components ($_{lm.st}$, and $\phi.st, r\phi.st$, respectively).

Figure 4.3 illustrates the main differences between the baseline system and our supertagged system. As shown in part (A) of the figure, the baseline employs a five-gram language model over English words. In part (B), our supertagged system employs a five-gram model over supertags and feature functions of supertagged phrase-pairs. Furthermore, we add smoothing feature functions (as shown in part (C)) with statistics over supertagged phrase-pairs where we marginalize over supertagged target phrases by using five-grams on supertags without the lexical items.

4.5.2 Language Models with a Grammaticality Factor

As is usual with *n*-gram language models, the probabilities of the supertagged language model are smoothed to provide better estimation for unobserved sequences. While smoothing the *n*-gram statistics is essential, the language model may prefer less grammatical supertag sequences over more grammatical ones. Recall that CCG supertags encode valency and directionality information for the arguments; this information can be used to construct an ‘almost parse’ with the help of external CCG composition operators. We are interested in examining the effect of applying the combinatory operators on the supertags sequence, such that we measure the grammaticality of the sequence based on the number of violated operators. We opt to examine this effect by integrating a penalty term into the language model which expresses the extent to which the formal composition operators are violated in a sequence of supertags. In general, the kind of violations that can arise between two

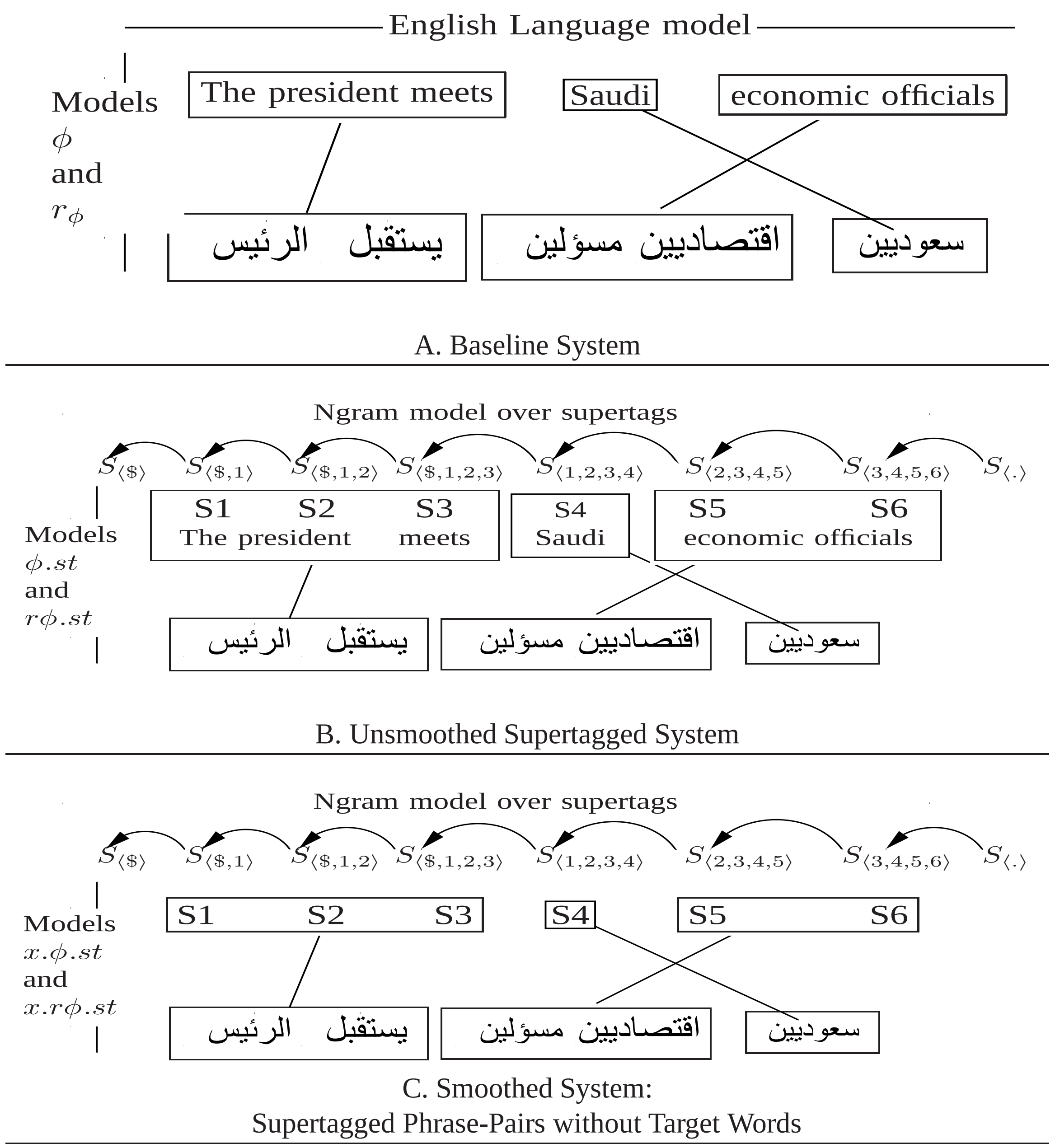

English Language model
Models ϕ and r_ϕ
The president meets
Saudi
economic officials
يستقبل الرئيس
اقتصاديين مسؤلين
سعوديين
A. Baseline System
Ngram model over supertags
$S_{\langle\$\rangle}$ $S_{\langle\$,1\rangle}$ $S_{\langle\$,1,2\rangle}$ $S_{\langle\$,1,2,3\rangle}$ $S_{\langle1,2,3,4\rangle}$ $S_{\langle2,3,4,5\rangle}$ $S_{\langle3,4,5,6\rangle}$ $S_{\langle.\rangle}$
Models $\phi.st$ and $r\phi.st$
S1 S2 S3
The president meets
S4
Saudi
S5 S6
economic officials
يستقبل الرئيس
اقتصاديين مسؤلين
سعوديين
B. Unsmoothed Supertagged System
Ngram model over supertags
$S_{\langle\$\rangle}$ $S_{\langle\$,1\rangle}$ $S_{\langle\$,1,2\rangle}$ $S_{\langle\$,1,2,3\rangle}$ $S_{\langle1,2,3,4\rangle}$ $S_{\langle2,3,4,5\rangle}$ $S_{\langle3,4,5,6\rangle}$ $S_{\langle.\rangle}$
Models $x.\phi.st$ and $x.r\phi.st$
S1 S2 S3
S4
S5 S6
يستقبل الرئيس
اقتصاديين مسؤلين
سعوديين
C. Smoothed System:
Supertagged Phrase-Pairs without Target Words

Figure 4.3: Supertagged PBSMT system (from (Hassan et al., 2008a))

consecutive supertags can be characterized as “type-mismatches” between the argument that one supertag expects to its left (right) and the supertag actually occurring to its left (respectively right).

Ideally, the more violations, the less grammatical the sequence of supertags is deemed to be according to the grammar. This is due to the observation that these violations represent non-satisfied dependency relations in this context. It is not to be taken for granted, though, that the formal grammaticality criteria (measured here in terms of compositionality) should coincide with better translation output or a better grammatical sequence.

The penalty factor that we experiment with here is added as a feature in the log-linear model, although we do not tune this parameter, instead relying on the supertag LM feature weight. The penalty term measures the ratio of the number of encountered violations over all adjacent supertag pairs to the total number of adjacent pairs in a sequence of supertags. For a supertag sequence of length L, which has V operator violations (as measured by the CCG system), the language model P will be adjusted to become $P*$ where $P* = P \times (1 - \frac{V}{L-1})$.

This term is, of course, no longer a simple, smoothed maximum-likelihood estimate of a language model, nor is it a true probability. Nevertheless, this mechanism provides a simple, efficient integration of a global compositional (grammaticality) measure into the *n*-gram language model over supertags.

As illustrated in Figure 4.4, the sentence with a possible supertag sequence (not correct sequence). The sentence length is six words, i.e. $L = 5$ operator applications in total over pairs of adjacent supertags. The supertag of *“believes”* demands directly to its right an (NP) with Forward Application; however, it finds a (PP/NP) instead. This counts as a single violation $V = 1$, since all other pairs of adjacent words have supertags that match under Forward and Backward application. Note that the supertag that fits best in the given sequence for *“believes”* is $(S\backslash NP)/PP$, which would be appropriate for a sentence such as *“He believes in me”*.

While measuring the grammaticality of a target language sentence by penalizing the

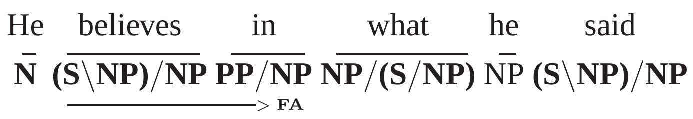

He believes in what he said
N (S\NP)/NP PP/NP NP/(S/NP) NP (S\NP)/NP
> FA

Figure 4.4: A grammatical violation example.

supertags LM, based on the number of grammatical violations, might be viewed as somewhat *ad hoc*, it has provided us with better insight into the usability of combinatory operators for our models. As a matter of fact, our observations from using this method have reformed our research agenda for the rest of this thesis, as we will discuss in the next chapter.

4.5.3 Supertagged Phrase-based Decoder

The decoder used in this chapter is Moses (Koehn et al., 2007), a log-linear decoder similar to Pharaoh (Koehn, 2004a), modified to accommodate supertag phrase probabilities and supertag language models. It is worth noting that while Moses implements factored translation models, in this work we do not avail of this functionality. In our preliminary results (Hassan et al., 2006), we built a decoder using the MOOD framework (Patry et al., 2006). After the development of Moses, we switched to it as it is much faster than MOOD framework.

4.6 Experiments

4.6.1 Arabic–to–English

In this section we evaluate the effect of lexical syntax on translation quality. A number of experiments were carried out on the NIST open domain news translation task from Arabic–to–English, with the aim of examining the effect of incorporating both supertagging approaches (CCG or LTAG) in our models with varying data sizes.

Data and Settings

The experiments were conducted for Arabic–to–English translation and tested on the NIST 2005 evaluation set. The systems were trained on the LDC Arabic–English parallel corpus; we used the news part (130K sentences, about 5 million words) to train systems with what we call the *small* data set, and the news together with a large part of the UN data (2 million sentences, about 50 million words) for experiments with *large* data sets.

The *n*-gram target LM and the supertag LM were built using 250M words from the English GigaWord Corpus using the SRILM toolkit (Stolcke, 2002).[1] This differs from our previous work in (Hassan et al., 2006), where just 25M words of the English GigaWord Corpus was used for building both target LMs. For the LTAG supertags experiments, we used the most recent LTAG English supertagger[2] (Bangalore et al., 2005) to tag the English part of the parallel data and the monolingual LM data. This supertagger is a MaxEnt supertagger employing more than 5000 different supertags; in the large data that we supertagged (more than 300M words), we encountered 3994 different supertags. For the CCG supertag experiments, we used the CCG supertagger of (Clark and Curran, 2004) and the 'C&C' tools[3] to tag the English part of the parallel corpus as well as the CCG supertag LM data.

The NIST MT03 test set was used for development, particularly for optimizing the interpolation weights using Minimum Error Rate Training (MERT) (cf. Section 2.4.4) using the Moses scripts. As we described in Section 2.4, the baseline system deploys 6 log-linear features, while our Supertagged Phrase-based system (Section 4.5) added 5 more features. Thus our system has to tune 11 features using MERT. We found that MERT was not able to tune this relatively large number of features in one batch; thus we resorted to running MERT in several batches trying to tune a subset of the parameters at each time, i.e. tuning translation parameters in a batch and then fix them and tune the language model parameter in the next batch and so on. While we realize that this is not

[1]http://www.speech.sri.com/projects/srilm/

[2]This supertagger employs a more elaborate supertag set than the original supertagger employed in (Hassan et al., 2007b).

[3]http://svn.ask.it.usyd.edu.au/trac/candc

the best solution for this problem, but we had to deal with it in this *ad hoc* manner. As we discussed in Section 2.6.1 this is a real limitation of MERT estimation for log-linear models as has just been highlighted in (Chiang et al., 2008).

Baseline System

The baseline system is a state-of-the-art Phrase-based SMT system as described in Section 2.4. Our baseline uses GIZA++[4] (Och and Ney, 2003) to obtain word-level alignments in both language directions. The bidirectional word alignment is used to obtain phrase translation pairs using heuristics presented in (Och and Ney, 2003). More specifically, we use the intersection of the bidirectional GIZA++ alignments and Grow-Diag heuristics to expand the alignments by adding direct neighbour and diagonal neighbour alignment points (cf. Section 2.4.5 for more details). The Moses framework (Koehn et al., 2007) is used for phrase extraction and decoding.[5]

We built two baseline systems with two different-sized training sets: 'Base-SMALL' (5 million words) and 'Base-LARGE' (50 million words) as described in the previous section. Both systems use a 5-gram language model with Kneser-Ney discounting (Kneser and Ney, 1995; Goodman, 2001) built using 250 million words from the English Giga-Word Corpus. Table 4.1 presents the BLEU scores (Papineni et al., 2002) for both systems on the NIST 2005 MT Evaluation test set.

System	**BLEU Score**
Base-SMALL	40.08
Base-LARGE	44.18

Table 4.1: Baseline systems' BLEU scores

Note that these scores (especially the latter) are indicative of quite good quality systems already.

[4]http://www.fjoch.com/GIZA++.html
[5]http://www.statmt.org/moses/.

Baseline vs. Supertags on Small Data Sets

We compared the translation quality of the baseline systems with the LTAG and CCG supertags systems (LTAG-SMALL and CCG-SMALL). The results are given in Table 4.2.

System	**BLEU Score**
Base-SMALL	40.08
LTAG-SMALL	42.52
CCG-SMALL	41.74

Table 4.2: LTAG and CCG systems on small data

All systems were trained on the same parallel data. The LTAG supertag-based system outperforms the baseline by 2.44 BLEU points absolute (or 6.1% relative), while the CCG supertag-based system scores 1.66 BLEU points over the baseline (4.1% relative). These statistically significant improvements (using bootstrap resampling (Koehn, 2004b)) indicate that the rich information in supertags helps select better translation candidates.

POS Tags vs. Supertags

A supertag is a complex tag that localizes the dependency and the syntactic information from the context, whereas a normal POS tag just describes the general syntactic category of the word without further constraints. In this experiment we compared the effect of using supertags and POS tags on translation quality. As can be seen in Table 4.3, while

System	**BLEU Score**
Base-SMALL	40.08
POS-SMALL	40.73
LTAG-SMALL	42.52

Table 4.3: Comparing the effect of supertags and POS tags

the POS tags help (by 0.65 BLEU points, or 1.7% relative increase over the baseline), they clearly underperform compared to the supertag model (by 4.4% relative).

The Usefulness of a Supertagged LM

In these experiments we studied the effect of the two added feature (cost) functions: supertagged translation and language models. We compared the baseline system to the supertags system with the supertag phrase-table probability but without the supertag LM. Table 4.4 shows the performance of the baseline system (Base-SMALL), the LTAG system without supertagged language model (LTAG-TM-ONLY) and the LTAG-SMALL system with both supertagged language and translation models. The results presented

System	**BLEU Score**
Base-SMALL	40.08
LTAG-TM-ONLY	41.46
LTAG-SMALL	42.52

Table 4.4: The effect of supertagged components

in Table 4.4 indicate that the improvement is due to a shared contribution between the supertagged translation and language models: adding the LTAG TM improves the BLEU score by 1.38 points (3.45% relative) over the baseline, with the LTAG LM improving BLEU score by a further 1.06 points (a further 2.65% increase).

Scalability: Larger Training Corpora

Outperforming a Phrase-based SMT system on small amounts of training data is less impressive than doing so on really large data sets. The issues here concern scalability as well as the question as to whether the Phrase-based SMT system is able to bridge the performance gap with the supertagged system when reasonably large sizes of training data are used. To this end, we trained the systems on 2 million sentences of parallel data, deploying LTAG supertags and CCG supertags. Table 4.5 presents the comparison between these systems and the baseline trained on the same data. The LTAG system improves by 1.82 BLEU points (4.1% relative), but the CCG system gives an even larger increase: 1.91 BLEU points (4.3% relative). While the relative improvement score for CCG is a little higher than with the smaller data set, for LTAG it is slightly lower (6.1%

on the smaller data set). Nonetheless, the fact that sustained increases are to be found at all is probably due to observing more data with different supertag contexts, which enables the models to select better target language phrases. The difference between the LTAG system and the CCG system is statistically insignificant.

System	**BLEU Score**
Base-LARGE	44.18
LTAG-LARGE	46.00
CCG-LARGE	**46.09**

Table 4.5: Performance on large training data

Adding a grammaticality factor

As described in Section 4.5.2, we integrate a grammaticality factor based on two standard CCG combination operations, namely Forward and Backward Application, and Forward Composition. Table 4.6 compares the results of the baseline, the CCG with an *n*-gram LM-only system (CCG-LARGE) and CCG-LARGE with this 'grammaticalized' LM system (CCG-LARGE-GRAM). We see that bringing the grammaticality tests to bear on the supertagged system gives a further improvement of 0.79 BLEU points, a 1.7% relative increase, culminating in an overall increase of 2.7 BLEU points, or a 6.1% relative improvement over the baseline system.

System	**BLEU Score**
Base-LARGE	44.18
CCG-LARGE	46.09
CCG-LARGE-GRAM	**46.88**

Table 4.6: CCG with grammaticality factor (CCG-LARGE-GRAM)

Combining LTAG and CCG Supertags

A natural question to ask is whether LTAG and CCG supertags are playing similar (overlapping, or conflicting) roles in practice. Using an oracle to choose the best output of the two systems gives an average per-sentence BLEU score of 44.1, indicating that the

System	BLEU Score
CCG-Large	41.03
LTAG-Large	40.87
LTAG-CCG-Oracle	44.10

Table 4.7: Sentence average BLEU score for CCG, LTAG and oracle with both

combination provides significant room for improvement (cf. Table 4.7). However, our efforts to build a system that benefits from the combination did not give any significant performance change. We investigated two issues that might lead to this: the interpolation mechanism, and the conflict between LTAG and CCG constraints.

We tried different ways of interpolating LTAG and CCG models; the LTAG-CCG system in Table 4.8 uses log-linear interpolation while LTAG-CCG2 uses additive interpolation by averaging both LTAG and CCG scores before interpolating with other systems components. Both systems results fall below that of the baseline system. System LTAG-CCG3 deploys both a CCG language model and a CCG translation model, but it uses only the LTAG translation model. The score of LTAG-CCG3 is somewhat better than the baseline, but remains lower than both the LTAG and CCG scores when deployed separately. Obviously, more sophisticated ways of combining the two could result in better performance than a simple interpolation of the components.

System	BLEU Score
Base-LARGE	44.18
LTAG-LARGE	46.00
CCG-LARGE	46.09
LTAG-CCG	41.81
LTAG-CCG2	42.86
LTAG-CCG3	44.93

Table 4.8: Sentence average BLEU score for CCG, LTAG and oracle with both

Conflicts between LTAG and CCG constraints may lead to such effect, given the need to satisfy different, and possibly contradicting constraints. In any case, this experiment indicates that combining constraints from different grammatical formalisms should be done with care. Figure 4.5 demonstrates an example where LTAG and CCG outputs contradict. Both the LTAG and CCG systems preferred *authorities reported that* rather than the baseline *the authorities that*, but the LTAG and CCG constraints conflict on the subphrase *allowed ... family ...* This may have resulted from obtaining bad subphrases when combining both approaches.

Source:: وافادت السلطات انهَا سمحت له بالاتصال هاتفيًا بافراد عائِلته من السيارة المصفحة التي كان فيهَا

Reference: *The authorities said he was allowed to contact family members by phone from the armored vehicle he was in.*

Baseline: *the authorities that it had allowed him to communicate by phone with his family of the armored car where*

LTAG: *authorities reported that it had allowed him to contact by telephone with his family of armored car where*

CCG: *authorities reported that it had enabled him to communicate by phone his family members of the armored car where*

LTAG+CCG: *authorities reported that it had allowed him by telephone contact his family of the car of armored personnel where*

Figure 4.5: Conflict of CCG and LTAG when combined

Systems Output Analysis

In order to acquire a deeper insight into the effect of the supertag components on system output, as well as where they might not help, we conducted a manual analysis of a subset of the system's output against the baseline and reference translations. To select interesting cases, we employed a threshold (20 BLEU points) as the minimal difference between the sentence-level BLEU score of the CCG-LARGE system and that of the Base-LARGE system. There are only 41 cases where (Base-LARGE $-$ CCG-LARGE)≥ 20. From the 76 cases where (CCG-LARGE $-$ Base-LARGE)≥ 20 we randomly sampled 50 sentences. We inspected both sets of cases manually against the reference translation, with the aim of finding an explanation as to why supertags improved over the baseline and vice versa. Naturally we tried to find a mutually exclusive classification of the test cases. Where this was not possible we employ a general bucket called “Other reasons”.

N=50 test sentences		
Reason	#	%
Inserting verb omitted by baseline	11	22%
Better reordering	11	22%
Better word/phrase selection	5	10%
Other reasons	23	46%

Table 4.9: How CCG improves over baseline

Table 4.9 exhibits the reasons for improved output for the CCG-based system over the baseline system. Only 22% of the cases are due to improved reordering, mainly verb/subject and noun/adjective, as illustrated in Figure 4.6. The CCG system correctly includes a verb which was omitted by the baseline system in 22% of cases; this concerns verbs such as *said, concluded, is, signed*, etc., as shown in Figure 4.7.

Source:: من جانبه ، قال ايفانوف ان العلاقات الروسية - الصينية شهدت تقدمًا ملحوظًا خلال الاعوَام الماضية

Reference: *For his part, Ivanov said that Sino-Russian relations have undergone marked progress in recent years.*

Baseline: *For his part , said Ivanov that russian-chinese relations witnessed a remarkable progress during the past years .*

CCG: *For his part , Ivanov said that russian-chinese relations witnessed a remarkable progress during the past years .*

Figure 4.6: Improved Reordering in the CCG system

Source:: واجرَى عنان تحقيقًا داخليًا في فبراير لكنه الغاه في مارس تمهيدًا لتحقيق اوسع ومستقل

Reference: *Annan opened an internal investigation in February but cancelled it in March in preparation for a broader, independent investigation.*

Baseline: *Annan was to internally in February but abolished in March as a prelude to broader and independent .*

CCG: *Annan conducted an internal inquiry in February but abolished in March in preparation for broader and independent .*

Figure 4.7: Overcoming missing verbs in the CCG system

Omitting verbs turns out to be a problem for the baseline system (see Figure 4.8). Both supertagged systems have a more grammatically strict language model than a standard word-level Markov model, and so exhibit a preference (in the CCG system especially) for the insertion of a verb with a similar meaning to that contained in the reference sentence. We think that the improvement of restoring omitted verbs is due to the fact that verbs have rich supertag structures that encode full syntactic information and therefore can directly influence the system to opt for a more syntactic output.

Source: واكد الجانبان علي دور منظمة التجارة العالمية
Reference: *The two sides highlighted the role of the World Trade Organization*
Baseline: *The two sides on the role of the World Trade Organization (WTO)*
LTAG: *The two sides on the role of the World Trade Organization*
CCG: *The two parties reaffirmed the role of the World Trade Organization*

Figure 4.8: Baseline system omits verbs while supertags system can produce them

Apart from improvements with these verbs, the CCG system achieves better output due to improved word/phrase selection in about 10% of cases. In a large number (48%) of cases, the CCG system improvement is accounted for by a number of reasons, e.g. selecting the correct form of verb (cf. Figure 4.9, where we see 'killed' vs. 'killing', among other improvements), restoring negation (cf. Figure 4.11), improved grammaticality (cf. Figure 4.12), and a variety of other reasons (cf. Figure 4.10). Although restoring the negation may not be the direct effect of supertags unlike the case with restoring verb since the negations do not have rich supertag structures, we think that such improvements are due to the fact that we are using a log-linear model with a variety of features; A small change in the cost of any of these features may influence the system to produce a better translation.

Table 4.10 shows the reasons as to why the baseline system gives improved output compared to the CCG-based system. In 14.6% of cases, the output of the CCG system reads better than the baseline and conveys the correct meaning, yet the baseline matches the reference translation more closely. Another 12.1% of the cases concern long NPs and PPs for which supertaggers do not offer a good treatment; CCG tends to prefer briefer translations in such cases. In another 7.3% of instances, the CCG system inserts extra

Source:: واسفر النزاع عن مقتل ١٠٥ مليون شخص علَى الاقل ونزوح حوالي اربعة ملايين شخص

Reference: *This dispute has killed at least 1.5 million people and displaced approximately four million people.*

Baseline: *The conflict on the 1.5 killing of at least a million people and the displacement of some four million people .*

CCG: *The conflict killed at least 1.5 million people and the displacement of about four million people .*

Figure 4.9: Improved Verb Forms in the CCG system

Source: وَاسفرت هذه الاعتداءات عن سقوط اكثر من مِئة قتيل منذ ايار / مايو ٢٠٠٣.

Reference: *These attacks have resulted in over 100 deaths since May 2003.*

Baseline: *In the attacks left more than 2003 people killed since May .*

CCG: *Resulted in these attacks on more than one hundred deaths since May 2003 .*

Figure 4.10: Improved translation in general in the CCG system

Source: محمود عباس: الجدار والمستوطنات لن توفر الامن لاسرَائِيل

Reference: *Mahmoud Abbas: The Wall And Settlements Will Not Bring Israel Security*

Baseline: *Mahmoud Abbas , the wall and settlements will provide security to Israel*

CCG: *Mahmoud Abbas : the wall and settlements will not provide security for Israel*

Figure 4.11: Restored negation in the CCG system

Source: الرباط ١-١٤ (ا ف ب) يدور جدل حاد في المغرب حول حرية الصحافة في مَا يتعلق بالمواضيع التي تمس الملك محمد السادس شخصيًا بعد نشر مقالات تنتقد مداخيل العاهل المغربي ونشاطاته

Reference: *Rabat 1-14 (AFP) - A sharp debate is raging in Morocco on the freedom of the press with regard to matters connected personally to King Mohamed VI following the publication of articles criticizing the Moroccan monarch's income and activities.*

Baseline: *Rabat 14-1 (afp) - was a sharp controversy in morocco on press freedom in terms of topics affecting king Mohamed VI himself after publishing articles critical of the revenues of the moroccan .*

CCG: *Rabat 14-1 (afp) - a sharp controversy in Morocco on press freedom in respect of topics affecting king Mohamed VI personally after the publication of articles criticizing the moroccan monarch revenues.*

Figure 4.12: Better syntactic modelling in the CCG system

function words (e.g. “of”, “that”, “which”) which are not available in the reference translation. In another 7.3% of the cases, the CCG supertagger was confronted with an Out-Of-Vocabulary item which lead to a deterioration in supertagging output. Verb confusion (5.1%), where one verb is nested under another (e.g. “said” and “returned” in “He said that life has almost returned to normal”), also constitutes a problem for the CCG-based system, as it tries simultaneously to satisfy the argument specifications of both verbs, which are often incompatible.

N=41 test sentences		
Reason	#	≈%
Better CCG output that matches less with reference	6	14.6%
Long NPs and PPs	5	12.1%
CCG wrongly inserting function words	3	7.3%
Supertagger facing OOV	3	7.3%
Verb-confusion	2	5.1%
Other reasons	22	53.6%

Table 4.10: How Baseline improves over CCG

In general we observed that supertagging seems to help most when the baseline system already has reasonable alternative translations; where supertagging improves the selection of a better translation. Whenever the baseline system forms a bad starting point (mostly when translation involves many short phrases), the CCG supertags do not help much; in fact, the CCG supertags may even lead to slightly worse output than the baseline in such cases. This analysis is not surprising for two reasons. Firstly, supertags offer a syntactic improvement over the baseline system mainly with regard to the grammaticality of the output via constructing *‘almost parsing’*. Secondly, when the input sentence consists of unseen combinations of words/phrases relative to the training data, the phrase-based systems perform the translation using the smallest phrases found in the training data (in the worst-case, word-to-word translation). In this case, the supertagged Phrase-based SMT helps a little as the translation candidates are not good enough to construct *‘almost parsing’*. It might be helpful to use an approach based on the confidence score of the baseline system such that we may be able to decide when a syntatctic model should be used or not.

While on average our system selects more grammatical output than the baseline, it is still limited to the same set of hypothesis translations that can be built by the standard reordering mechanism used in the baseline system. Nonetheless, in last year's IWSLT-07 evaluation, it was encouraging that our supertags-based Arabic–English system described in (Hassan et al., 2007a) was ranked first by some margin in the human evaluation, despite this clear advantage of more fluent output not carrying over to the automatic evaluation scores.

4.6.2 German–to–English

In order to examine the applicability of our method to other language pairs, we carried out a number of experiments on German–to–English. The data used was that of the shared task of the ACL 2007 MT Workshop (WMT 2007),[6] comprising over 1 million sentence pairs of Europarl (Koehn, 2005) and (much smaller, about 1 million words) news commentary data, giving a total of around 22 million words for each language.

The language models (both *n*-gram and supertag-based) were trained on the 39 million words of English monolingual data. The standard setup for the workshop was used to build the baseline system, and we built a CCG supertags system in the same manner as described in section 4.5. We used devset2006 (2000 sentences) for parameter tuning using Minimum Error Rate Training (MERT) (cf. section 2.4.4), and Testset-2006 for testing. Each of the test sets was composed of 2000 sentences.

The results are contained in Table 4.11:

System	**BLEU Score**
Baseline	27.07
CCG Supertags	27.55
Baseline (w/o Brevity Penalty)	27.34
CCG Supertags (w/o Brevity Penalty)	29.47

Table 4.11: CCG Supertags System for German–English

[6]http://www.statmt.org/wmt07/shared-task.html

Here we can see that the CCG supertags system improves on the baseline by 0.48 BLEU points, a 1.77% relative increase in performance. While at first glance this result might be seen as disappointing compared to the Arabic–English scores, there are a number of explanations for the relative discrepancies.

Firstly, for Arabic–English there were 4 reference translations for the MT05 testset against which the output sentences were evaluated, whereas for German–English, there exists just a single reference.

Secondly, the translation output from the CCG supertags system tends to be shorter in sentence length than the single reference and so is highly penalized by the brevity penalty in the BLEU metric (cf. Section 2.7). This can be seen by the final row of results in Table 4.11, where we observe an increase in BLEU score of 2.4 points, an 8.9% relative improvement, compared to the baseline performance, when the negative effects of the brevity penalty are disregarded.[7] In this light, an improvement of around 0.5 BLEU points, even taking into account the effect of the brevity penalty, is a good improvement for a single reference testset for any language pair.

Figure 4.13 provides a good example of the sorts of improvements which a supertag-enriched model of translation provides compared to a baseline Phrase-based SMT system. The supertagged model provided enhancements with respect to treatment of negation, reordering, better verb treatment and overall a more syntactic translation. The baseline wrongly omits "not", and does not capture the collocation "Mann report". It can be seen that the CCG system generates good verb strings ("is being completely forgotten"), and in general provides more fluent and intelligible output, even on this 49-word German sentence.

Figure 4.14 shows that just like for Arabic–English (cf. Figure 4.7), the tendency for SMT systems to omit verbs in translation is overcome when supertags are deployed. Firstly we see that "is to be" is correctly inserted in the subordinate clause, and also that in the relative clause, "which does not belong" appears to render the translation perfectly

[7]The baseline score has not been much affected by the brevity penalty as the translation is slightly shorter than the reference.

Source: *Ich habe nicht für den Bericht Mann gestimmt, denn bei allem tatsächlich notwendigen Streben nach Gleichbehandlung in Beschäftigung und Beruf braucht deswegen noch nicht im Übereifer so weit gegangen zu werden, dass der Schutz der Freiheiten und die Achtung des Rechtsstaates dabei völlig in Vergessenheit geraten.*

Reference: *I have not voted for the Mann report because, while it is indeed necessary to seek equal treatment for people in employment and occupation, it is also necessary to refrain from pushing zeal to the point of abandoning all protection of freedoms and all respect for the rule of law.*

Baseline: *I have voted in favour of the report because , in particular , man is actually needed quest for equal treatment in employment and occupation is therefore not yet in excess of zeal went so far as to say , the protection of freedoms and respect for the rule of law is completely forgotten .*

CCG: *I have not voted for the Mann report because , in fact , with all the necessary search for equal treatment in employment and occupation is therefore not yet gone so far in excess of zeal , that the protection of freedoms and respect for the rule of law is being completely forgotten .*

Figure 4.13: Improved performance of the CCG system for German–English.

intelligible. Note, of course, that this latter improvement does not perfectly match the reference, so will not receive the full benefit when it comes to an increase in BLEU score, despite being a perfectly acceptable translation.

Source: *Wenn die Richtlinie annehmbar und durchführbar sein soll, darf sie nicht mit Literatur und Wunschdenken überlastet werden, die in einem legislativen Text nichts zu suchen haben.*

Reference: *If the directive is to be adopted and implemented, it must not be encumbered with a literary approach and wishful thinking, which have no place in a legal document.*

Baseline: *If the directive acceptable and is going to be possible , it must not be overloaded with literature and wishful thinking , not in a legislative text .*

CCG: *If the directive is to be reasonable and workable , it must not be overloaded with literature and wishful thinking , which does not belong in a legislative text .*

Figure 4.14: Overcoming missing verbs in the CCG system for German–English

4.7 Conclusions and Open Questions

SMT practitioners have on the whole found it difficult to integrate syntax into their systems mainly because of the mismatch between the notions of an SMT phrase and a constituent in mainstream linguistics. The main difficulty lies in devising some syntactic structure that fits with phrases but does not admit (much) redundant ambiguity into the phrase translation table. Such redundancy leads to even larger tables, more complex probability models and less efficient decoding.

In this chapter, we have presented a novel model of Phrase-based SMT which integrates linguistic lexical descriptions, supertags, into the target language model and the target side of the translation model. Supertags fit seamlessly with Phrase-based SMT as they are lexical, linguistically rich and can be used in efficient Hidden Markov Models (Rabiner, 1989) as well as full parsing models. However, currently there are only a few languages for which supertag-sets and supertaggers exist, which limits the current applicability of our model to translation to such languages.

We believe that our use of supertags in the experiments conducted in this chapter exemplifies the importance of lexical syntactic information such as subcategorization frames for improved translation output. Much of this lexical information can be acquired without the need for full parsing or treebanking.

We have carried out extensive experiments on small and very large training and test sets for Arabic–English translation. While using LTAG supertags gives the best improvement over a state-of-the-art Phrase-based SMT system for the smaller data set, using CCG supertags works best on the large training set. Adding grammaticality factors based on algebraic compositional operators gives the best result, namely 46.88 BLEU, or a 6.1% relative increase over the baseline. This result compares very favourably with the best systems on the NIST 2005 Arabic–English task.

The experiments on very large training data are important because they provide evidence that ever increasing amounts of data (and correspondingly larger phrase-translation tables) will not bridge the performance gap with a system that incorporates syntactic information about phrase combination/ordering.

In addition, we demonstrated the applicability of our approach to another language pair, namely German–English. Our CCG supertags model improves over the baseline

Phrase-based SMT system by 0.48 BLEU points, a 1.77% relative increase. This is a satisfactory improvement when one takes into account that only one reference translation was available for this 2000-sentence testset. Nonetheless, when the severe effect of the BLEU brevity penalty is disregarded, we observed an increase in BLEU score of 2.4 points, an 8.9% relative improvement, compared to the baseline performance.

In this chapter we showed that integrating lexical syntax in the translation model and language model of a Phrase-based SMT system have caused translation quality to improve. We showed that a supertagged translation model provided improvements on its own, and more improvement was observed when used with the supertagged language model. Our analysis of the translation output showed that a very wide range of improvements were brought about by the use of a supertags-based system, including improved reordering, overcoming the tendency of SMT systems to omit verbs, improved verbal constructions, proper handling of negation, and well-formed syntactic output in general. In this regard, we noted that in a recent large-scale open evaluation, the output from our Arabic–English supertags-based system (Hassan et al., 2007a) was preferred by human evaluators, although given the remaining differences between the output and the reference translations, this does not always result in improvements in BLEU score.

Having addressed the question as to wheather lexical syntax can be of use in Phrase-based SMT; we now move our attention to the related questions of whether lexical syntax can provide Phrase-based SMT with full parsing capability and whether this is needed by Phrase-based SMT systems. We will try to answer these questions in the next two chapters.

Chapter 5

Incremental Dependency-based Language Modeling

5.1 Introduction

In Chapter 4, we described our supertagged Phrase-based SMT model which integrated a supertagged translation model and an *n*-gram supertagged language model into a baseline Phrase-based SMT system; this integration significantly improved the translation accuracy. Perhaps surprisingly, we also showed that adding simple heuristic grammaticality measures can further improve the translation accuracy. This unexpected improvement highlighted a drawback of supertagged language models; there is no guarantee that the sequence of proposed supertagged phrases constitutes a valid syntactic constituent. Another more serious, though expected, drawback is that supertagged language models cannot handle long-range dependencies. In this chapter, we introduce a solution for those problems: an incremental dependency-based language model that enables the seamless integration of incremental dependency parsing into Phrase-based SMT systems.

In this chapter, we introduce a novel Incremental Dependency-based Language Model (IDLM) using CCG incremental parsing. In Chapter 6, we will show how our proposed IDLM is integrated into the SMT model.

5.2 Incremental Dependency-based Language Model for MT

5.2.1 From Supertagged to Dependency-based Language Models

Lexical syntax deploys rich syntax descriptions —supertags— that match individual words, and a limited set of Combinatory Operators which are used to combine supertags into a set of constituents/parses. The supertagging language model replaces the set of Combinatory Operators with an *n*-gram language model over the sequence of supertags (thus *'almost parsing'*). Originally, *'almost parsing'* had been proposed for handling monolingual strings, where the given sequence of words already constructs a presumed syntactic structure (Bangalore and Joshi, 1999). In the bilingual —MT— case, the sequence of candidate target words might not construct a valid syntactic structure nor a compelling sequence of associated supertags; therefore, achieving *'almost parsing'* by deploying a supertagged *n*-gram language model on the huge space of hypotheses, representing the candidate translations, is more challenging in the MT case than in the monolingual parsing case.

We argue that the MT case needs a more sophisticated mechanism that can satisfy three important aspects. First, it needs to efficiently support long-range dependencies and construct full parse structures such that it would enable the MT system to distinguish between different translation candidates based on their role in constructing the parse structure and satisfying the syntactic dependencies. Second, as is widely known, Phrase-based SMT systems produce the translation candidates incrementally by processing source words from left-to-right in a Markov fashion; therefore, this mechanism should work in an incremental manner. Third, the mechanism should be computationally efficient such that it can be integrated into large-scale Phrase-based SMT systems.

In this chapter, we introduce an incremental dependency-based language model which deploys CCG incremental parsing mechanism to construct the parsing structure step-by-step, where each step represents the accumulation of parsing decisions as the parser *incre-*

mentally consumes the input word-by-word from left-to-right. The proposed dependency-based language model complies with the Markovian nature of Phrase-based SMT decoders; therefore, it has the potential to be integrated seamlessly with such systems. Furthermore, it is based on a deterministic parsing approach, i.e. it maintains a limited number of parse decisions at each parsing step which makes it very efficient computationally. In the next section we will briefly introduce our proposed Incremental Dependency-based Language Model (IDLM).

5.2.2 IDLM Overview

An incremental model of syntax and semantics construction was proposed in (Milward, 1994a). In this model the syntactic process is represented by a sequence of transitions between adjacent syntactic/semantic states. The syntactic representation is built step-by-step, hence incremental, from left-to-right while traversing the input string as shown in (5.1). The syntactic state contains all the dependency information about fragments that have already been processed so far. The parser produces fully connected intermediate structures *incrementally* while moving from one word to the next. As (Milward, 1994a) indicated the model can be seen as a Markov model with an unbounded number of states (in principle).

$$S_0 \xrightarrow{w_1} S_1 \xrightarrow{w_2} S_2 \cdots\cdots S_{n-1} \xrightarrow{w_n} S_n \tag{5.1}$$

Before we go further in describing our proposed IDLM, let us first clarify some notions regarding incremental parsing, left-to-right parsing and lookahead. Incremental dependency parsing is the process of constructing the dependency graph step-by-step, so that at each step the constructed partial graph is never altered or revised in any later step. On the other hand, the construction of the incremental dependency graph does not have to be strictly left-to-right. In fact, it can be left-to-right, right-to-left or even bidirectional as long as the incrementality condition mentioned above is maintained. Moreover, incremental parsers may have access only to a fixed, limited window of lookahead words. In other words, an incremental parser may delay decisions but may not delay decisions indefinitely, i.e. requiring lookahead for the whole sentence. Incremental parsers without lookahead information decide what is the next expansion to the dependency graph without access to any words to the right of the current word. By contrast, parsers with lookahead have access to a limited number of words to the right of the current word. As a matter of fact, the lookahead is equivalent to buffering a number of words before processing them; as stated by (Marcus et al., 1983), a deterministic parser can buffer and examine a small number of words before adding them to the existing structure. Based on these definitions, we call an incremental, left-to-right parser without lookahead information a *fully incremental parser*, while we call an incremental left-to-right parser with limited lookahead capability a *weakly incremental parser*. It is worth noting that fully incremental parsers are cognitively plausible (Marslen-Wilson, 1973; Sturt and Lombardo, 2004), while weakly incremental parsers can serve well for syntax-based language modeling where a context of the word is usually provided for scoring.

Our IDLM is an embodiment of the theoretical representation outlined above, where we use an incremental parser based on CCG as the grammatical representation of the syntactic/semantic states and the transition actions that lead from a state to another.

As shown in (5.2), each word w_i is associated with a lexical syntactic/semantic descriptor st_i. At each transition, a parsing action o_i is associated with that transition, which transforms the current parse-state S_i to the next state S_{i+1} which in turn represents a new partial syntactic derivation. When the last word is encountered, a final state S_n represents the final syntactic structure for the given sequence of words. Such a sequence of parsing actions constructs the parsing derivation step-by-step.

$$S_0 \xrightarrow[w_1,st_1]{o_1} S_1 \xrightarrow[w_2,st_2]{o_2} S_2 \cdots\cdots\cdots S_i \xrightarrow[w_i,st_i]{o_i} S_{i+1} \cdots\cdots\cdots S_n \tag{5.2}$$

We use incremental CCG as our grammatical representation such that the lexical descriptor st_i is represented by a CCG supertag, and the parsing action o_i is represented by a CCG Combinatory Operator with the state S_i being a composite CCG category. Each state S_i is determined exactly by the previous state S_{i-1} and a choice of a supertag st_i and an operator o_i. Therefore, the probability $P(W, S)$ of a word sequence W and associated final parse-state sequence S, which represents a possible derivation, can be described as in Eqn (5.3):

$$P(W,S) = \prod_{i=1}^{n} \overbrace{P(w_i|W_{i-1}S_{i-1})}^{Word\ Predictor} . \overbrace{P(st_i|W_i)}^{Supertagger} . \overbrace{P(o_i|W_i, S_{i-1}, ST_i)}^{Operator\ Tagger} \tag{5.3}$$

This probability represents the product of the production probabilities at each parse-state and is similar to the structured language model representation in (Chelba, 2000):

- $P(w_i|W_{i-1}S_{i-1})$ is the probability of w_i given the previous sequence of words W_{i-1} and the previous sequence of states S_{i-1}.
- $P(st_i|W_i)$ is the lexical descriptor (supertag st_i) probability given the word sequence W_i up to the current position. This is represented by a sequence tagger (supertagger) in our CCG incremental parser.
- $P(o_i|W_i, S_{i-1}, ST_i)$ represents the parsing action (operator o_i) probability given the previous words, supertags and state sequences up to the current position. This is represented by a sequence operator tagger in our CCG incremental parser.

It is worth noting that the proposed language model parser is deterministic, in the sense that it maintains a limited number of parse-states (only one here) that represent possible parsing decisions at each word position. This characteristic is very important for incorporating IDLM into large-scale MT systems due to its computational efficiency.

In this chapter we discuss in detail the our work introduced in (Hassan et al., 2008b). In the remainder of this chapter, we will describe the mechanics of this incremental parser, while the deployment of IDLM in MT systems will be introduced in Chapter 6.

In Section 5.3, we will review the related work on syntax-based language models as well as incremental parsing. In Section 5.4 we will introduce our incremental parsing approach in detail. In Section 5.5, we will describe the transformation of the CCGbank derivations into the incremental derivations needed for the incremental parser. In Section 5.6, we will describe the implementation details of our incremental parser. We will present our parser evaluation in Section 5.7 and finally provide some discussion in Section 5.8.

5.3 Related Work

Many psycholinguists have claimed that the meaning of a sentence can be obtained before all words in the utterance have been heard (e.g. (Marslen-Wilson, 1973; Sturt and Lombardo, 2004)). Incrementality in parsing has also been proposed for real-time applications such as speech-to-speech translation, where analysis of the input utterance needs to be updated on a regular basis. In this section, we will discuss the differences between our proposed IDLM and the previously proposed syntactic language model approaches (cf. Section 3.6). Then we will review related work introduced for incremental parsing in general.

5.3.1 Syntax-based Language Models

In Section 3.6, we reviewed previous work that incorporated syntactic language models into speech recognition systems and MT systems such as (Chelba, 2000; Charniak, 2001; Roark, 2001; Wang et al., 2004; Xu et al., 2002; Collins et al., 2005). All these approaches were evaluated only on small-scale speech recognition tasks. As for MT, only (Charniak et al., 2003) integrated the model proposed in (Charniak, 2001) into a syntax-based MT system (Yamada and Knight, 2001). All the previous approaches depend on non-deterministic techniques to grow a huge number of partial derivations which is unmanageable for large-scale applications such as MT or speech recognition. This has limited the usability of these approaches to very small tasks and/or re-ranking of system output. Another major aspect is that the previous approaches deploy CFG or dependency grammar that cannot handle non-constituent constructions in Phrase-based SMT systems (cf. Section 2.4).

Our proposed IDLM differs from this related work in four major respects:

- It is based on incremental parsing that seamlessly matches the incremental nature of SMT decoders.
- It is deterministic, in the sense that it maintains a limited number of parse-states that represent possible parsing decisions at each word position. This characteristic is very important for incorporating IDLM into large-scale MT systems due to its computational efficiency.
- The grammatical representation is based on CCG structures which enable the handling of non-constituent constructions.
- The parser seeks out intermediate connected structures, unlike previous approaches which deployed dependency relations or head words to enable syntax-based probabilities into the language model.

5.3.2 Incremental Parsing: Related Work

As we are using incremental parsing for our IDLM, we will review here the most relevant work for incremental parsing. Most current parsers do not tackle the problem of sentence analysis in an incremental fashion. State-of-the-art parsers such as (Collins, 1999) and (Charniak, 2000) require the derivation of a packed parse forest via dynamic programming, prior to a probabilistic disambiguation of the full sentence. As the packing of the parse forest is largely non-deterministic, incrementality is not an option here.

In contrast, partial parsers such as (Abney, 1991) do not output a full sequence of connected phrases, which causes the constraint of incrementality to fail for a quite different reason.

(Nivre, 2004) suggests that deterministic dependency parsing (e.g. (Yamada and Matsumoto, 2003)) is an intermediate solution between full and partial parsing, in that the building of a full parse of the input string is the aim, while at the same time remaining robust, efficient and deterministic.

(Nivre, 2004) describes an incremental approach to deterministic dependency parsing. While strict incrementality was not possible using his framework, as far as well-formed utterances are concerned, the degree of incrementality which is achievable approaches 90%.

(Ratnaparkhi, 1997) proposed a linear time model based on Maximum Entropy framework to determine chunks and higher syntactic structures; however he used multiple passes over the input string. Based on (Nivre, 2004) and (Ratnaparkhi, 1997), (Sagae and Lavie, 2006) introduced a statistical shift-reduce parser that uses a probabilistic framework to determine the shift and reduce actions and keep multiple possible parse decisions that are handled by a beam strategy.

(Shen and Joshi, 2005) use the term 'semi-incremental' to refer to parsers (both left-corner (e.g. (Collins and Roark, 2004)) and head-corner (e.g. (Yamada and Matsumoto, 2003))) which permit multiple iterations of left-to-right scans, rather than just one.

In contrast to these models, (Shen and Joshi, 2005) introduce an approach for fully incremental parsing of spinal Lexicalized Tree Adjoining Grammar (LTAG), which supports full adjunction, a dynamic treatment of coordination, as well as non-projective dependencies. (Shen and Joshi, 2005) observe that their model of incremental parsing with LTAG is very closely related to the supertagging approach of (Bangalore and Joshi, 1999), except that while supertagging can be seen as a two-stage approach (supertagging and composition of the complete derivation via the elementary trees), they incorporate the supertagger and dependency analyser dynamically in a similar way to (Bangalore, 2000). While the work described in (Shen and Joshi, 2005) has much in common with the approach proposed in this chapter, such as using supertagging and using classifications techniques to assign the parsing actions, there remain two significant differences that limit the capability of using this parser in language modeling. Firstly, (Shen and Joshi, 2005) requires full access to the sentence and thus may delay parsing decision indefinitely. Secondly, the parser uses a stack of disconnected derivations to represent the left context similar to (Xu et al., 2002), which further complicates its usage for language modeling.

Incremental parsing was applied to Categorial Grammar in (Milward, 1995) using a state-transition (or dynamic) processing model, where each state consists of a syntactic type together with an associated semantic value. In (Milward, 1994a), a generic approach for dynamic syntax and incremental parsing is proposed based on an infinite state Markov representation.

The model of incremental parsing for CCG that we propose here is largely inspired by ideas presented in (Milward, 1995), (Bangalore and Joshi, 1999) and (Bangalore, 2000), in that we use a state-transition model, based on an infinite state Markov representation, using CCG supertags and learning the parsing actions at each step. We describe our approach in the next three sections, together with experiments demonstrating the effectiveness of this method.

5.4 Incremental Parsing for CCG

In this work, the incremental parsing process is represented by an infinite Markov model. A parsing derivation is built step-by-step, where the words represent the transitions between states, and each state represents the partial parsing derivation constructed so far. Furthermore, each state is associated with a composite CCG category such that the number of possible states is (in principle) unbounded. The complex CCG category defines the required arguments at the current state, while the partial parsing derivation represents the partial dependency interpretation constructed so far.

The incremental parsing process consists of the construction of such dependency graphs in a step-by-step manner. At each state the partial dependency structure can be

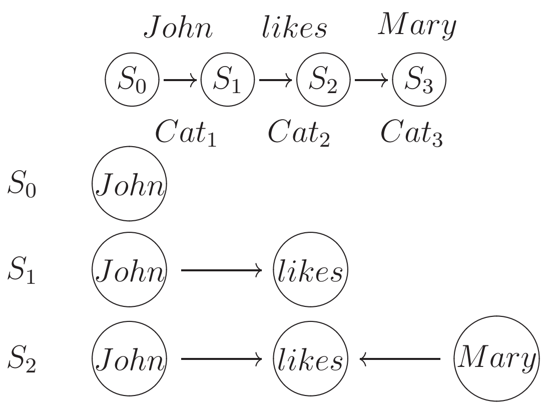


Figure 5.1: Illustration of the incremental parser representation and the associated intermediate dependency graphs at each state.

represented as a directed graph with nodes representing words and arcs representing dependency relations. Given a string of words, a sequence of partial interpretations and associated dependency graphs can be established. When the last word in the sentence has been processed, the graph represents the dependency structure of the whole sentence. It is worth mentioning that the model presented here is not restricted to fully connected graphs, i.e. during parsing, the proposed incremental parser can quite naturally handle partially connected graphs.

Figure 5.1 illustrates the incremental parsing representation. At the initial state S_0, the dependency graph is simply the node representing the first word *"John"*. The transition to the next state S_1 is triggered by the verb *"likes"*, where the dependency graph associated with state S_1 shows the realized dependency between *"likes"* and *"John"*. Finally the last word triggers the final state, and the parser is able to construct the full dependency graph which is associated with the last state S_3. Each state is associated with a complex CCG category, *Cat1*, *Cat2* and *Cat3* respectively.

The proposed approach deploys three modules in a cascade: (1) a statistical *Supertagger*, (2) a statistical *Operator tagger*, and (3) a deterministic *Parsing State Realizer*.

Figure 5.2 illustrates the operation of the cascaded architecture. First the supertagger assigns a possible supertag sequence to the words, shown under the words. Second, the operator tagger assigns a sequence of left-to-right operators, shown on the arrows' heads, which are able to satisfy the required dependency structure. Finally, the deterministic state

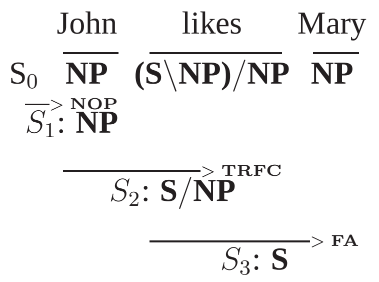


Figure 5.2: A sentence and a possible supertag sequence, operator sequence and state sequence. NOP: No Operation; TRFC: Type-Raise Forward Composition; FA: Forward Application.

realizer constructs the parse-states and the associated dependency graph incrementally, using the two assigned sequences.

The supertagger and the operator tagger have to be trained on left-to-right incremental CCG derivations. In order to obtain such data, we transformed the CCGbank (Hockenmaier and Steedman, 2007) from normal form derivations to strictly left-to-right derivations that can satisfy the dependencies in the CCGbank. The next section presents the transformation technique that we developed to obtain the appropriate training data.

Figure 5.3-a illustrates the transformation and training phase of the incremental parser; the CCGbank with associated dependency structures is transformed into two sequences of supertags and operators. The supertags sequence is used to train a MaxEnt supertagger and the operators sequence is used to train a MaxEnt operator tagger; this is described in detail in Section 5.6.1.

Figure 5.3-b illustrates the runtime parsing operation of the incremental parser; the supertagger and the operator tagger are used in a cascade to assign appropriate supertag and operator sequences to the given sentence. Both supertag and operator sequences are fed into the state realizer to construct the incremental parsing and the corresponding dependency graph step-by-step. This is described in detail in Section 5.6.2.

5.4.1 Merits of CCG for Incrementality?

We present a novel approach for wide-coverage incremental parsing based on CCG. As we described in Chapter 3, there are currently two supertagging approaches: LTAG (Joshi and

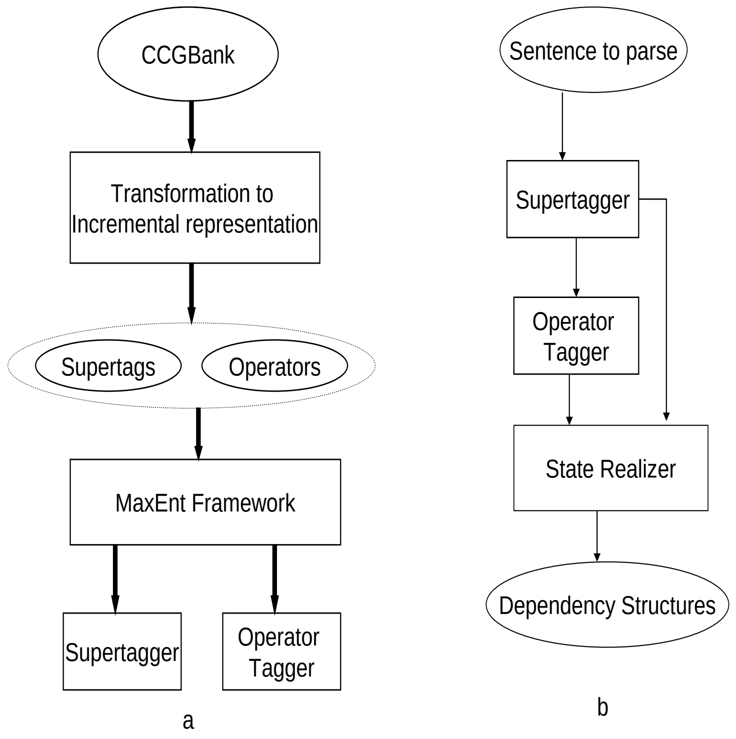


Figure 5.3: Incremental Parser: a: Transformation & Training phase, b: Parsing runtime phase.

Schabes, 1991) and CCG (Steedman, 2000). The two approaches have more similarities than differences (cf. Section 3.5); however, our proposed incremental parsing approach deploys CCG for several reasons:

- CCG (Steedman, 2000) is a lexicalized grammatical theory where the CCG lexical entries define syntactic categories which encode syntactic valency and directionality; these categories can be augmented by a semantic representation to provide compositional semantics with a completely transparent interface between surface syntax and logical semantics. Although in this thesis we focus on syntactic structures, CCG provides the possibility of expanding the proposed approach to a semantic representation as well [1].

- The CCG Combinatory Operators assemble lexical entries together into derivation-trees; each partial or complete syntactic derivation corresponds directly to a structure. For example, strings such as *"John likes"* have a natural interpretation as

[1]cf. Chapter 3 for detailed discussion on the compositional semantics capability of CCG.

constituents. (Doran and Bangalore, 1994) highlighted that the flexibility of CCG derivations allows the handling of non-constituent constructions that LTAG cannot handle, which is due to the fact that LTAG trees represent rigid structures while CCG categories allow more flexibility in the derivation process. Unlike many other linguistic theories, this flexibility gives CCG an advantage over other grammatical formalisms in handling non-constituent constructions for both incremental parsing and Phrase based SMT with arbitrary phrase boundaries (cf. (Tillmann and Xia, 2003; Koehn, 2004a)).

- As highlighted in (Steedman, 2000), CCG can represent every leftmost string as a constituent even if it is not a syntactic constituent. This can enable any left branching (left-to-right) parser to work fully incrementally.

- A fully incremental dependency parser is only possible if the leftmost graph is fully connected at each parse state, which has been highlighted in (Nivre, 2004). This is only possible with grammars like CCG where the type raising and compositional capabilities can be utilized to keep the graph connected even when not resolving a dependency relation.

- CCG has a wide-coverage treebank available, the CCGbank (Hockenmaier and Steedman, 2007).[2] The CCGbank is a CCG transformation of the Penn Wall Street Journal Treebank (Marcus et al., 1993); obtained by transforming the parse trees into normal form derivations in CCG. The CCGbank provides a wide-coverage CCG lexicon together with head-dependency annotations; therefore, it could be used to obtain the data needed for our proposed incremental parser.

We present a linear-time, incremental CCG parser. Our approach (Section 5.4) is based on a representation of parses as a sequence of parse-states, each representing the accumulation of parsing decisions as the parser consumes the input word-by-word from left-to-right. A parse-state is constructed by applying a CCG Combinatory Operator to the

[2]CCGbank is available through LDC, Catalog No.: LDC2005T13.

$$\begin{array}{cccc} \text{I} & \text{gave} & \text{them} & \text{advice} \\ \overline{\mathbf{NP}_1} & \overline{((\mathbf{S}\backslash\mathbf{NP}_1)/\mathbf{NP}_2)/\mathbf{NP}_3} & \overline{\mathbf{NP}_2} & \overline{\mathbf{NP}_3} \end{array}$$

Figure 5.4: CCG dependency structure.

previous state and the supertag of the current word. The parser constructs, incrementally, only a linear number of parse-states in sentence length. In the next section, we will define the CCG dependency structure as it is used in this work.

5.4.2 CCG Dependency Structure

CCG dependency structures consist of a set of CCG predicate-argument relations defined through the argument slots in the CCG lexical categories. Consider the sentence in Figure 5.4 where we see that the ditransitive verb has the category ($((\mathbf{S}\backslash\mathbf{NP}_1)/\mathbf{NP}_2)/\mathbf{NP}_3$) which encodes the dependency information of this verb. However, the dependency relations are established when a parsing derivation is constructed and the argument slots are filled with the appropriate categories. In this example, the first slot NP_1 of the verb *"gave"* is filled with the subject *"I"*, the second slot NP_2 is filled with the first object *"them"* and finally the third slot is filled with the second object *"advice"*. Thus we can interpret the dependency relations once the arguments are filled.

In this thesis, the CCG dependencies are used for two purposes. First, they are used to control the transformation process of the CCGbank from normal form derivations into incremental derivations. Second, they are used to evaluate the overall performance of our incremental parser by measuring how the parser can produce the dependencies.

5.5 Transforming the CCGbank into left-to-right Derivations

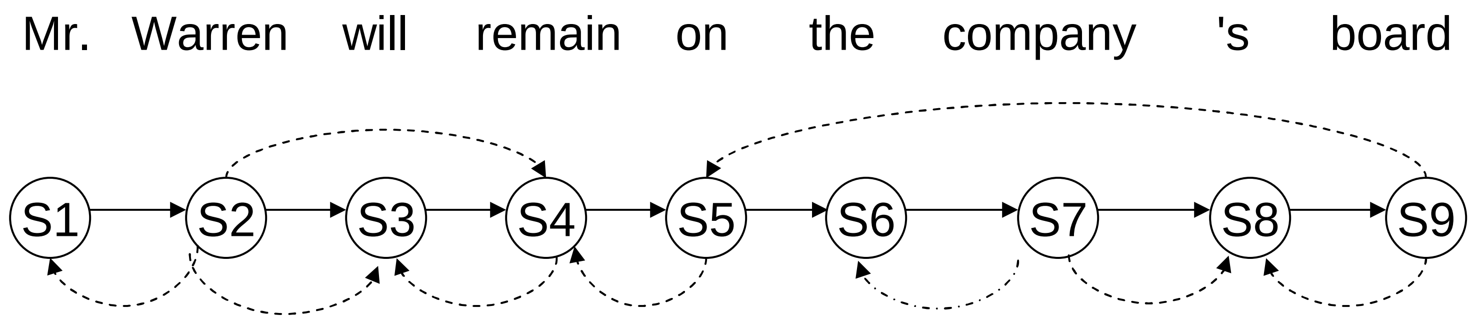
The main objective of the transformation process is to obtain training data annotated with supertags as well as a sequence of left-to-right operators such that we are able to satisfy the corresponding syntactic dependencies in the CCGbank.

For each sentence in the CCGbank, we apply the following procedure:

- Initialize empty operator sequence and empty unsatisfied dependencies.
- For each word:
  1. Add current dependencies to unsatisfied dependencies.
  2. Check unsatisfied dependencies:
     (a) If adjacent dependency with simple categories, then assign *application* operators;
     (b) If adjacent dependency with complex categories, then assign *composition* operators;
     (c) If long-range dependency, then apply *Type Raising* followed by *Forward Composition*.
  3. Handle special cases, if any:
     (a) Coordination cases (subsection 5.5.2),
     (b) Apposition and interruption (subsection 5.5.3),
     (c) WH-movement (subsection 5.5.4),
  4. Update Current state,
  5. Assign selected operator to the operator sequence.
  6. Update the dependencies by removing satisfied dependencies.

This procedure deploys the dependencies available in the CCGbank in order to assign the simplest possible operator sequence that is able to satisfy, and reproduce, the dependency structure of the sentence under investigation.

Figure 5.5 illustrates the transformation process, step-by-step, on a sentence of the CCGbank. At the beginning of the process, we start with the words, the associated supertags and the dependency relations, indicated by curved dotted arrows in the figure.



Supertag	NP/NP	NP	(S\NP)/(S\NP)	(S\NP)/PP	PP/NP	NP/NP	NP	(NP/NP)\NP	NP
Operator	NOP	FA	TRFC	FC	FC	TRFC	FA	FC	FA
State Cat.	NP/NP	NP	S/(S\NP)	(S/PP)	(S/NP)	(S/(NP\NP))/NP	(S/(NP\NP))	(S/NP)	S

Figure 5.5: Illustration of the CCGbank transformation process into incremental derivations (from (Hassan et al., 2008b)).

The purpose of the transformation process is to induce the state sequence and the operator sequence. This operator sequence along with the supertag sequence should be able to reproduce the given dependency relations.

The transformation process proceeds word-by-word, and at each word position we check all previous and current unsatisfied dependencies. The transformation proceeds as follows:

1. State *S1* is an initial state; therefore, it will be associated with operator *NOP*, which performs no operation, and the state category will be equivalent to the current word category *NP/NP*.

2. Moving to State *S2*, we first check the current and previous dependencies. In this case, there is a dependency between the word in first position, *"Mr."*, and the word at the current position *"Warren"*, shown by dotted arrows in the figure. As this dependency relation is adjacent and in the forward direction then the Operator *FA* is associated with this transition and so the state is transferred to *S2* with category *NP*.

3. Moving to State *S3* is triggered by the word *"will"*, which has both backward and forward dependencies. Therefore, the operator *TRFC* (Type-Raise and Forward Composition) is applied to fulfill the backward dependency and the potential forward dependency as well.

4. Moving to State *S4* is triggered by the word *"remain"* which is linked with the word *"will"* by a forward dependency relation; therefore, a Forward Composition *FC* operator is assigned. The state becomes *(S/PP)*, which indicates a requirement for a prepositional phrase to the right.

5. Moving to State *S5* is triggered by the word *"on"* which is linked to the previous verb *"remain"* and hence a Forward Composition *FC* operator is assigned changing the state to *(S/NP)*. This state indicates a dependency that requires a noun phrase to the right in order to be satisfied.

6. Moving to State *S6* is triggered by the word *“the”* which has neither backward nor forward dependencies; however, it is linked through a chain of dependencies with a future position which satisfies the current open dependency, the word *“board”*. Therefore, we apply the *TRFC* operator to type-raise the current word to the required dependency category and then perform a forward composition.

7. Moving to State *S7* is triggered by the word *“company”* which has a forward dependency with the previous position; therefore, the *FA* operator is applied.

8. Moving to State *S8* is triggered by the word *“’s”* which has adjacent forward and backward dependencies; therefore the FC operator is applied. This changes the state to *(S/NP)* which indicates that a noun phrase is required to satisfy the previous dependency.

9. Moving to State *S9* is triggered by the word *“board”* which is linked back to the word *“on”* at state *S5*. A simple FA operator is finally applied to construct the complete sentence category *S*.

The above illustration shows how the CCGBank is transformed. We started with a supertag sequence and a dependency graph, and ended with the corresponding operator and state sequences. However, the same procedure applies during parsing, i.e. if we have the supertag sequence and the operator sequence then we can construct the incremental states and the dependency graph step-by-step as we showed.

Certain more complex cases need special handling; therefore, we added some special operators to handle them, namely for coordination, cases of apposition and interruption, and WH-movement. These new operators together with the other operators used in the parser are described in the next section.

5.5.1 Incremental Combinatory Operators

Table 5.1 presents each operator used in our incremental parser, together with the percentage of its usage in the transformed CCGbank. It is clear that the simple, standard operators

Operator	**Description**	**Usage %**
FA	Forward Application	34.7
FC	Forward Composition	24.4
NOP	No Operation	16.1
BA	Backward Application	13.1
TRFC	Type Raising + Forward Composition	4.0
BC	Backward Composition	2.8
COORD	Coordination	2.4
INTR	Interrupters	1.6
WHMV	WH-movement	0.9

Table 5.1: Operators' Utilization.

of CCG are much more widely used than the more complex operators introduced in our method.

Our proposed set of Combinatory Operators are binary operators whose two arguments are the previous state and the current supertag. When the operator is applied to its two arguments, the result is the current state category. For example in (5.4), an operator *FA* is applied to *State*$_2$ and *Supertag*$_3$ to produce *State*$_3$.

$$\begin{array}{ccc} \text{Supertag}_1 & \text{Supertag}_2 & \text{Supertag}_3 \\ \text{State}_1 & \text{State}_2 & \text{State}_3 \\ & \multicolumn{2}{c}{\overline{\qquad\qquad\qquad}{>}\,\text{FA}} \end{array} \tag{5.4}$$

We extended the set of standard CCG operators reviewed in Section 3.4.1 with new operators to handle various needs raised by the incremental nature of the parser. In this section, we will discuss in detail the newly introduced operators.

No Operation (NOP)

The operator *(NOP)* performs no operation on any two constituents, such that the resulting state remains the same as the previous one. *NOP* is used at the initial position when commencing the incremental parsing process, and is also used with some of the punctuation marks that do not alter the parse-states or the dependencies.

Type Raising Forward Composition (TRFC)

Type raising and forward composition act together to capture long-range right-side dependencies. We designed the presented incremental parser to push forward the needed dependencies by increasing the “eagerness” of the states’ categories. In other words, we push the dependencies forward such that they are always represented by the current state category. Our incremental parser achieves this eagerness by using Type Raising followed by Forward Composition.

(Steedman, 2000) defines Type Raising as a unary rule and Forward Composition as a binary combinatory rule. However, our incremental parser is restricted to binary operators; therefore, we combined type raising and forward composition in one operator called $TRFC$.

If a constituent with category X/Y is immediately preceded by a constituent with category Z such that X/Y has a long-range dependency on the right side to a category $Y\backslash Z$, Type Raising is used to raise the category Z to category Y and then forward composition is applied to push the required dependency forward.

Examples (5.5 and 5.6) show TRFC in action, where the subject NP is type-raised to S and then forward composed with $(S\backslash NP)/NP$) to compose (S/NP).

$$\begin{array}{cc} \mathbf{X} & \mathbf{(Y\backslash X)/Z} \\ \hline \multicolumn{2}{c}{\mathbf{X/Z}} \end{array} {>_{\text{TRFC}}} \tag{5.5}$$

$$\begin{array}{cc} \text{He} & \text{bought} \\ \mathbf{NP} & \mathbf{(S\backslash NP)\ /NP} \\ \hline \multicolumn{2}{c}{\mathbf{S\ /NP}} \end{array} {>_{\text{TRFC}}} \tag{5.6}$$

5.5.2 Coordination

Coordination constructions occur in a significant number of sentences in written text. Cognitive studies (Sturt and Lombardo, 2004) have highlighted the fact that coordination is an incremental operation; thus, we should be able to handle coordination efficiently within the proposed incremental parsing approach. Unfortunately, the CCGbank uses a simple category for coordination *conj* instead of the more elaborate category *(X\X)/X* which was originally defined for coordination in CCG (Steedman, 2000). The simple *conj* operator is not efficient for incremental parsing, because it does not provide any information on the coordinated elements. Therefore, we used the dependency information to assign more elaborate coordination categories to the coordinator. For example, if the coordination is performed on two noun phrases, the coordination category would be *(NP\NP)/NP*. Furthermore, we have added a new coordination operator (*COORD*) to handle these constructions in the Parsing State Realizer.

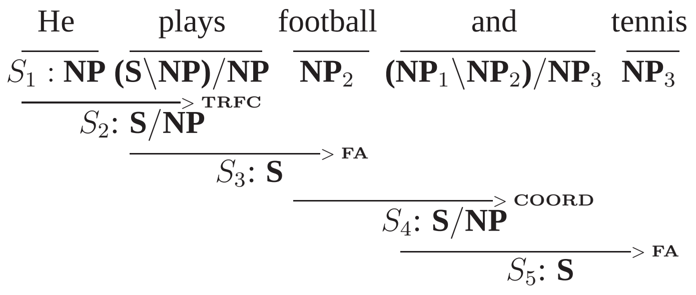


Figure 5.6: Coordination Handling.

The example shown in Figure 5.6 illustrates the handling of coordination during parsing. The conjunction “and” is associated with a supertag $(NP_1\backslash NP_2)/NP_3$ which indicates a coordination between two *NPs*.[3] The left argument NP_2 will be satisfied with the word *“football”*, while the right argument will be filled by the word *“tennis”*. This will construct a coordinated constituent NP_1 with the phrase *“football and tennis”*. During parsing, at the transition from $S3$ to $S4$ a coordination operator, *COORD*, is encountered, which causes the current state $S4$ to be a replica of state $S2$ with structure *S/NP*, i.e. expecting an *NP* to the right. In this way, the coordinated constituent NP_3 *“tennis”* is

[3]The categories subscripts in all the examples are for illustration purposes only.

expected on the right side to be coordinated with the previous *NP “football”*.

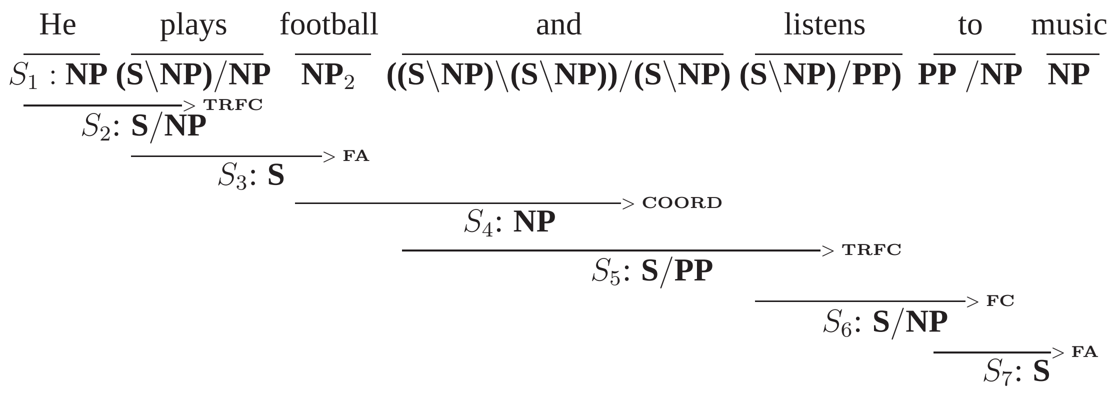


Figure 5.7: VP Coordination Handling.

Another example shown in Figure 5.7 illustrates the handling of coordination for two verb phrases (VP). In this example, the conjunction “and” is associated with a more complicated supertag $((S\backslash NP)\backslash(S\backslash NP))/(S\backslash NP)$ which indicates a coordination between two *VPs*. At the coordination state $S4$, a coordination operator, *COORD*, performs the coordination by producing a new state with structure *NP*, i.e. expecting a *VP* to the right. In this way, the coordinated *VP* constituent will be expected, just as the first *VP* was expected after $S1$.

Although the representation presented above could theoretically support non-constituent coordination (Milward, 1994b), the current implementation of our incremental parser does not support that.

5.5.3 Apposition and Interruption

Neither the CCGbank nor the WSJ treebank distinguish between the appositive comma and the coordination comma (Hockenmaier and Steedman, 2007). The comma mostly has a single supertag in the CCGbank that does not indicate its actual role in the syntactic structure. We adopted the syntactic patterns introduced in (Bayraktar et al., 1998) to identify the different possible syntactic categories of the comma. Based on these syntactic patterns, we enriched the supertags associated with the comma to indicate the correct syntactic role for the coordination, apposition and interruption cases.

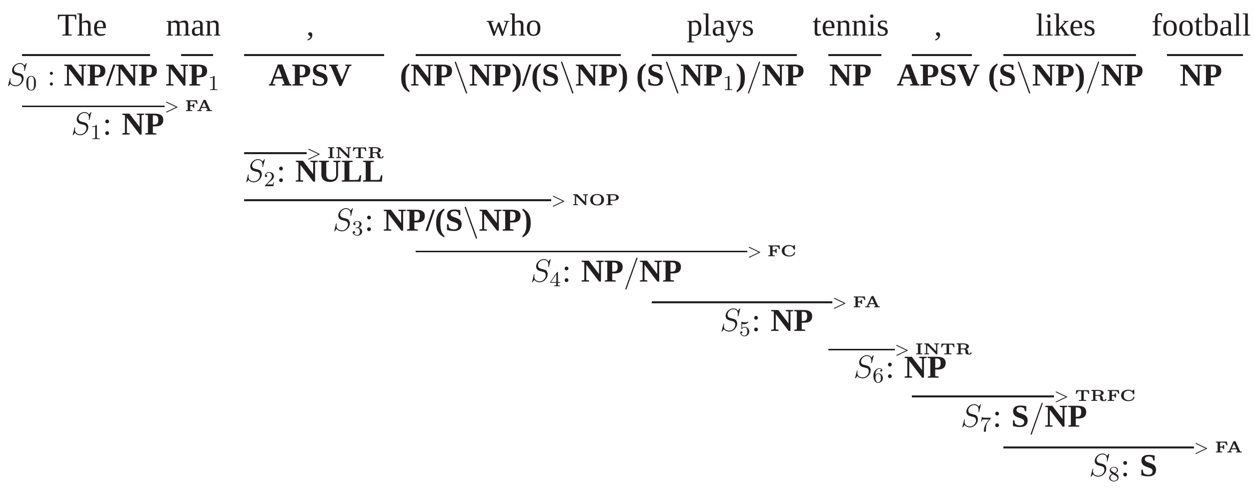

The man , who plays tennis , likes football
S0 : NP/NP NP1 APSV (NP\NP)/(S\NP) (S\NP1)/NP NP APSV (S\NP)/NP NP
> FA
S1: NP
> INTR
S2: NULL
> NOP
S3: NP/(S\NP)
> FC
S4: NP/NP
> FA
S5: NP
> INTR
S6: NP
> TRFC
S7: S/NP
> FA
S8: S

Figure 5.8: Apposition Handling.

Furthermore, we have added a new supertag and a new operator for handling such cases. The new supertag, APSV, is used for indicating apposition cases for commas and some other punctuation marks, such as bracketing. The operator *INTR* has been added to handle both interruptions and apposition.

The example in Figure 5.8 illustrates the handling of apposition during parsing. The parser consumes a noun phrase *"The man"* up to state S_1 then a comma with *APSV* supertags and operator *INTR* is encountered. The parser handles the apposition by moving to a *NULL* state S_2 and storing the interrupted state S_1; then the apposition phrase *"who plays tennis"* is consumed up to state S_6. At the transition from S_5 to S_6, a second apposition comma is encountered, so the parser terminates the apposition states and moves to S_6 which is equivalent to the interrupted state S_1. In this way, parsing of the sentence can continue from where it was interrupted; thus the NP *"The man"* will fill the subject argument of the verb *"likes"*.

5.5.4 WH-movement

WH-movement is a syntactic phenomenon where a syntactic category is required on the right but, having moved, is available only on the left. Consider the sentence in Figure 5.9, the verb *"sold"* has the category $(S\backslash NP_1)/NP_2$, i.e. it is a transitive verb, where if a subject NP_1 is available to its left, and an object NP_2 to its right, a sentence will have been

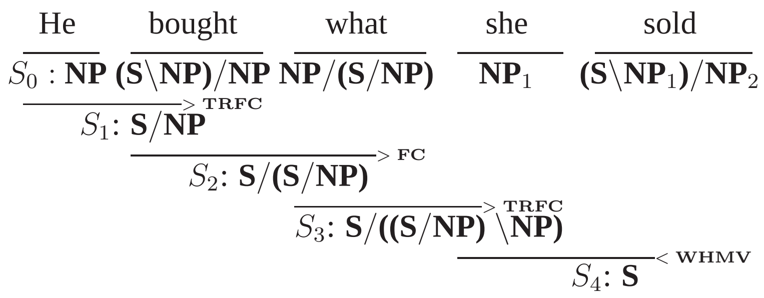


Figure 5.9: WH-movement Handling.

formed. The required object *NP*$_2$ *"what"* has already moved to an appropriate position somewhere to the left. Accordingly, we added a new operator *WHMV* to handle such cases of WH-movement in the incremental parsing framework. The *WHMV* operator reverses the direction of the arguments such that the parser seeks the object of the verb *"sold"* to the left instead of the right, such that a sentence is composed as shown in the example.

Having described the combinatory operators of our incremental parser, we will describe the parser's components in the following section.

5.6 Implementation Details of the Incremental Parser

5.6.1 Supertagger and Operator tagger

The transformed data from the CCGbank was used to train two Maximum Entropy (MaxEnt) classifiers: a supertagger, and an operator tagger. As shown in Eqn. (5.7), MaxEnt classification associates a weight λ_i with each feature function $\phi_i(Y, X)$. The weights are estimated during training in order to maximize the likelihood of the training data. MaxEnt can be used for sequence classification by converting the classification scores into probabilities and then using standard dynamic programming (Viterbi search). We train our MaxEnt model using sequential conditional generalized iterative scaling (Goodman, 2002). This method is a simple variation of Generalized Iterative Scaling (Berger et al., 1996), but converges faster by training the model parameters sequentially rather than simultaneously.

$$Y^* = \arg\max_Y P(Y|X) = 1/Z \quad exp \sum_i \lambda_i \phi_i(Y, X) \tag{5.7}$$

For the supertagger MaxEnt classifier, we use words and POS features with a window of two words to the left and two words to the right of the current word (hence it is considered 'weakly' incremental). For the operator tagger, we do not use any lexical features, but rather the POS and supertag features within the same window as the supertagger.

5.6.2 Parse-State Realizer

The *parse-state realizer* is a deterministic module that deploys the sequences of supertags and CCG incremental operators to realize the parse-states as well as the intermediate dependency graphs between words. The state realizer carries out the CCG operations incrementally and enables the special handling of coordination, apposition, interruption and WH-movement as described above.

The parse-state realizer constructs the dependency graph step-by-step by constructing intermediate dependency graphs word-by-word. The realizer performs the following steps for each word starting from a null state at the first word:

- Apply the current operator to the previous state and the current supertag,
- Change the current state to the new resulting state,
- Add edges to the dependency graphs between words that were linked as CCG arguments,
- Repeat until the last word has been processed.

Figure 5.10 illustrates the realizer operation along with the incrementally constructed partial dependency graphs at each state. At the initial state S_1, the Null Operator (*NOP*) is applied to the previous state, a Null state, and the current supertag *NP*; the resulting state

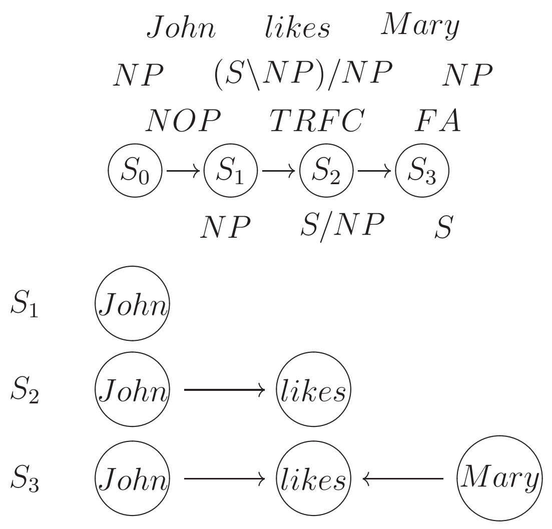


Figure 5.10: Illustration of the operation of the incremental parse-state realizer and the associated intermediate dependency graphs at each state.

is *NP* and the resulting dependency graph is simply the node representing the first word *John*. The transition to the next state S_2 is triggered by the verb *likes*, where the operator (*TRFC*) is applied to the previous state and the current supertag, resulting in a new state *S/NP*, which indicates that a further NP is needed on the right to compose a complete sentence structure. The dependency graph associated with state S_2 shows the realized dependency between *likes* and *John* which has resulted from the previous composition operation. Finally the last word triggers the final state, and the realizer is able to construct the full dependency graph associated with the last state S_3.

It is worth mentioning that the state realizer is the complement of the transformation process described in Figure 5.5. If we have both the supertag and operator sequences, then we are able to construct the state sequence and the corresponding dependency graph accordingly.

5.7 Experiments and Results

This section details a number of experiments carried out to test the effectiveness of the supertagger, the operator tagger, and our ability to capture the necessary dependencies

Architecture	**Lookahead**	**Search**	**Dependency Accuracy**	**Supertagging Accuracy**	**Operator Accuracy**	**Incremental**
Joint	NO	LOCAL	56.02	67.47		Incr.
	NO	GLOBAL	56.13	68.31		Semi
	YES	LOCAL	82.17	84.34		Incr.+LH
	YES	GLOBAL	83.20	85.02		Semi+LH
Cascaded	NO	LOCAL	59.01	68.11	76.19	Incr.
	NO	GLOBAL	59.30	68.62	76.53	Semi
	YES	LOCAL	86.31	91.62	90.76	Incr.+LH
	YES	GLOBAL	86.70	91.70	90.90	Semi+LH

Table 5.2: Supertagger, Operator tagger and Dependency results (F-Score) of all systems.

using a range of incremental parsers. We used the same data split as in (Clark and Curran, 2007). Sections 02–21 were used for training, section 00 for dev-testing of intermediate taggers, and section 23 for testing dependencies.

5.7.1 Supertagging Results

Given our introduction of new supertags for coordination, apposition, interruption, and WH-movement, we used section 00 to evaluate our supertagger's accuracy compared to the standard CCGbank set. Although our supertags are more complex, we obtain an F-score of 91.7 (cf. Table 5.2, last row, 'Supertagging' column), which compares favourably with the supertagger of (Clark and Curran, 2007), which scores 92.39 on the same dataset. Our supertags set is much richer than the supertags set of (Clark and Curran, 2007); therefore the results may not be directly comparable. While we have not carried out significance testing at this stage, it is clear that there is little difference between the two sets of scores, indicating that our supertagger is robust as well as accurate. As will be seen for all experiments in this section, this is only true when lookahead is utilised; note that our best score of 91.7 dips to 68.62—an absolute drop of 23.08 points, or a 33.6% relative decrease in performance—when lookahead is turned off.

5.7.2 Operator Tagging Results

In Table 5.2 we also present the results for our Operator tagger. This displays a very high accuracy (90.9%, cf. last row, 'Operator Tagging' column) even when no lexical features are used. We also contemplated a hypothetical situation in which we feed the correct (gold standard) previous syntactic state as a feature to the system. In this scenario an operator tagging score of 99.22% (8.32% absolute improvement, or 9.15% relative) was obtained, indicating that a high gain is to be expected if this state were to be made available to the operator classifier.

5.7.3 Dependency Results

In Table 5.2 we also present the results for unlabeled dependency accuracy using our method. We use the same evaluation criteria as (Clark and Curran, 2007) by comparing the dependency output of the incremental parser with the predicate-argument dependencies in the CCGbank. Testing on section 23 of the WSJ, we obtain an F-score of 86.7 (last row, 'Dependency' column). The score with the gold standard POS and supertags in the input is 87.5, 0.8% absolute (or 0.92% relative) higher than the result when using the POS, supertags and operators hypothesized by the system, but not by much. This overall result is considerably below the result reported in (Clark and Curran, 2007) (91.65% unlabelled dependency F-score). However, using a non-incremental bottom-up parser is much less efficient than our (weakly) incremental parser. (Clark and Curran, 2007) observe that on section 23 of the WSJ, while the parser of (Collins, 1999) takes 45 mins to parse all the sentences, and that of (Charniak, 2000) takes 28 mins, their parser takes just 1.9 mins. By contrast, our parser takes just 11 seconds, a speed-up of around ten times, on the same specification machine.

5.7.4 Cascaded vs. Joint Approach

The results reported above demonstrate the accuracy of the cascaded approach using two cascaded taggers: the first for supertags, and the second the operator tagger followed by the deterministic state realizer. In this section we compare the cascaded model with a joint model, where we train a single classifier that produces the supertags and operators simultaneously in the same step. In Table 5.2 we give the unlabeled dependency results for section 23 for the cascaded and joint models side-by-side for comparative purposes. The cascaded model significantly outperforms the joint model (by 3.5% absolute, or 4.2% relative; this rises to 4.3% absolute, or 5.17% relative, if we compare the joint model with the dependency score using the gold standard POS and supertags, as described in the previous section). Besides data sparseness, the joint model makes the choice of an operator at a certain position in the sentence based on supertag information only to the left of the current position because the joint model must guess supertag–operator pairs at once.

Note that our Cascaded version with lookahead and GLOBAL search is the semi-incremental model of (Shen and Joshi, 2005). They report an F-score of 89.3 on section 23 using a semi-incremental approach, together with extra information from Propbank (Palmer et al., 2005). While not directly comparable, we consider our performance to be on a par with theirs, with a considerable improvement in parsing time (they report a speed of 0.37 sent./sec.).

5.7.5 Effect of Lookahead

The present parser is just two words of lookahead away from being fully incremental. Here, we examine the effect of lookahead features on the supertagger, operator tagger and dependency results. We examine two versions of a supertag- and operator-classifier, namely a weakly incremental and a fully incremental version. The weakly incremental version deploys features in a window of two words to the left and two words to the right of the focus word. The fully incremental parser deploys features in a window of two words to the left only.

Looking at all the results in Table 5.2, the scores for the weakly (semi-)incremental versions of the parser barely differ from their fully incremental counterparts, whether we are concerned with dependency, supertagging or operator accuracy; the scores are higher, on the whole, but not by much.

By contrast, what is *extremely* significant is the extent to which lookahead is utilised. For all accuracy measures, huge improvements are to be seen when the parser avails of lookahead. Clearly, full incrementality at this stage comes at a high cost in accuracy, relative to the weakly incremental version, without any benefit in efficiency.

5.7.6 Examples

In this section we will guide the reader through three examples and examine the parser output. The examples are selected from newswire data typically used in MT evaluation tasks. The newswire data is harder to parse than the Penn Wall Street Journal Treebank (Marcus et al., 1993) data, which we have used for training and testing our parser. For the three examples shown, starting with the sentences, our incremental parser proceeded by tagging them with POS tags, supertags, operators and then the state realizer was applied.

Example 1

The example shown in Figure 5.11 demonstrates an incremental parsing output of the proposed parser. In the example shown here each state represents a partial construction of the dependency graph.

The example demonstrates how the parser is able to handle long-range dependencies and coordination. We will highlight some important aspects here:

- At state S_4, the parser assigned a *TRFC* operator, although the previous state S_2 has a required argument *NP* to the right. The more straightforward action is to fill this open argument with the noun phrase *"President Putin"* using a forward composition operator. However, this is not the correct dependency indicated by the sentence structure because *"They listened to the point of view"* not to *"President Putin"*. The parser can capture that effect by taking into account features from the word *('s)* a few positions ahead. This exemplifies how Markov-based syntax representations are able to capture long-range dependencies.

- At state S_{13}, the parser opts for a *NOP* operation for this comma which reflects that it has no apposition or coordination role.

- The coordination at state S_{21} is coordinating two long clauses, the first running from S_{16} to S_{20} and the second running from S_{22} to S_{28}. The *COORD* operator at S_{21} can restore the state back to a category similar to S_{15}, i.e. expecting the second noun phrase.

Example 2

The example shown in Figure 5.12 exemplifies the handling of apposition together with some other interesting issues.

- An apposition phrase runs between states *S5* and *S11*, which the parser indicated by assigning the *APSV* supertag and the *INTR* operator at both positions. This enables the parser to interrupt the normal sequence between those two states to construct the apposition noun phrase. After applying the *INTR* operator a new state sequence runs from *S6* up to *S10*. At state *S11* the interruption ends and the state becomes equivalent to the state at *S4* such that the state sequence is able to resume from where it was interrupted. In this way, the subject argument of the verb *"refused"* is filled with the word *"official"* which is eight positions away.

- Some intermediate states such as *S16*, *S19* and *S22* have a full sentence category *S*, which indicates that those are partially completed sentences for which all dependencies are satisfied.

State	Word	Supertag	Operator	State Category
S1	They	NP	NOP	NP
S2	listened	S\NP	BA	S
S3	to	((S\NP)\(S\NP))/NP	BC	(S/NP)
S4	President	NP/NP	TRFC	((S/(NP\NP))/NP)
S5	Putin	NP	FA	(S/(NP\NP))
S6	’s	(NP\NP)/NP	FC	(S/NP)
S7	point	NP	FA	S
S8	of	(NP\NP)/NP	BC	(S/NP)
S9	view	NP	FA	S
S10	on	(NP\NP)/NP	BC	(S/NP)
S11	various	NP/NP	FC	(S/NP)
S12	subjects	NP	FA	S
S13	,	,	NOP	S
S14	such	(NP\NP)/(NP\NP)	BC	(S/(NP\NP))
S15	as	(NP\NP)/NP	FC	(S/NP)
S16	human	NP/NP	FC	(S/NP)
S17	rights	NP	FA	S
S18	in	(NP\NP)/NP	BC	(S/NP)
S19	his	NP/NP	FC	(S/NP)
S20	country	NP	FA	S
S21	and	((NP\NP)/NP)	COORD	(S/NP)
S22	the	NP\NP	FC	(S/NP)
S23	latest	NP/NP	FC	(S/NP)
S24	crisis	NP	FA	S
S25	between	(NP\NP)/NP	BC	(S/NP)
S26	Russia	NP	FA	S
S27	and	((NP\NP)/NP)	COORD	(S/NP)
S28	Georgia	NP	FA	S
S29	.	.	NOP	S

Figure 5.11: Example1: Incremental Parsing.

State	Word	Supertag	Operator	State Category
S1	However	S/S	NOP	(S/S)
S2	,	,	NOP	(S/S)
S3	the	NP/NP	TRFC	((S/(S\NP))/NP)
S4	official	NP	FA	(S/(S\NP))
S5	,	APSV	INTR	NULL
S6	who	(NP\NP)/(S\NP)	NOP	(NP/(S\NP))
S7	requested	(S\NP)/(S\NP)	FC	(NP/(S\NP))
S8	to	(S\NP)/(S\NP)	FC	(NP/(S\NP))
S9	remain	(S\NP)/(S\NP)	FC	(NP/(S\NP))
S10	anonymous	S\NP	FA	NP
S11	,	APSV	INTR	(S/(S\NP))
S12	refused	(S\NP)/(S\NP)	FC	(S/(S\NP))
S13	to	(S\NP)/(S\NP)	FC	(S/(S\NP))
S14	give	(S\NP)/NP	FC	(S/NP)
S15	more	NP/NP	FC	(S/NP)
S16	details	NP	FA	S
S17	about	(NP\NP)/NP	BC	(S/NP)
S18	the	NP/NP	FC	(S/NP)
S19	negotiations	NP	FA	S
S20	in	(NP\NP)/NP	BC	(S/NP)
S21	which	NP/NP	FC	(S/NP)
S22	Cairo	NP	FA	S
S23	is	(S\NP)/(S\NP)	FC	(S/(S\NP))
S24	playing	(S\NP)/NP	FC	(S/NP)
S25	the	NP/NP	FC	(S/NP)
S26	role	NP	FA	S
S27	of	(NP\NP)/NP	BC	(S/NP)
S28	mediator	NP	FA	S
S29	.	.	NOP	S

Figure 5.12: Example 2: Incremental Parsing.

Example 3

State	Word	Supertag	Operator	State Category
S1	Its	NP/NP	NOP	(NP/NP)
S2	total	NP/NP	FC	(NP/NP)
S3	debt	NP	FA	NP
S4	was	(S\NP)/NP	TRFC	(S/NP)
S5	5	NP/NP	FC	(S/NP)
S6	trillion	NP/NP	FC	(S/NP)
S7	yuan	NP	FA	S
S8	,	,	NOP	S
S9	an	NP/NP	FC	(S/NP)
S10	increase	NP	FA	S
S11	of	(NP\NP)/NP	BC	(S/NP)
S12	5	NP/NP	FC	(S/NP)
S13	billion	NP/NP	FC	(S/NP)
S14	yuan	NP	FA	S
S15	,	((NP\NP)/NP)	COORD	(S/NP)
S16	or	conj	NOP	(S/NP)
S17	5	NP/NP	FC	(S/NP)
S18	percent	NP	FA	S
S19	from	(NP\NP)/NP	BC	(S/NP)
S20	nine	NP/NP	FC	(S/NP)
S21	months	NP	NOP	(S/NP)
S22	ago	((S\NP)\(S\NP))\NP	BA	NULL
S23	.	.	NOP	NULL

Figure 5.13: Example 3: Incremental Parsing.

In the example shown in Figure 5.13, the parser made a mistake by assigning a wrong operator at $S21$. It is worthwhile highlighting some issues here:

- The parser was able to construct partially completed sentences at states $S7$, $S14$ and $S18$.
- At state $S21$, the parser assigned a wrong operator NOP and this led to a wrong sequence of states up to the end of the sentence.
- The parser cannot construct a fully connected derivation. However, the partially connected derivation may still identify some correct dependencies.

5.8 Discussion

In this chapter we introduced our Incremental Dependency-based Language Model (IDLM) based on wide-coverage CCG incremental parsing. The introduced dependency-based LM has very interesting characteristics that facilitates its integration into Phrase-based SMT systems:

- The language model parser is deterministic in that it maintains a limited number of parsing decisions at each state which makes it very efficient for integration into large-scale Phrase-based SMT systems.
- It is incremental in Markovian fashion similar to Phrase-based SMT decoders.
- It can naturally handle non-constituent constructions, being based on CCG.
- The parser always seeks fully connected structures, not just using syntactic information to augment LM probabilities. At the same time, the parser can handle non-connected structures as well.
- The parser supports long-range dependencies and a number of interesting syntactic phenomena in a fully incremental left-to-right fashion.

It is worth mentioning that the current implementation of the incremental parser cannot be considered as a language model as it is, since this implementation employs a lookahead of words and a cascade of MaxEnt classifiers. However, this incremental parser can be used to parse training data with the incremental parsing information which could be used to train a language model to be used within SMT decoders to estimate the probability of a string-parse pair as described in IDLM formalization in Eqn (5.3).

As further work for the incremental parser itself, we think there are two main issues that could have a good effect on the parser's accuracy such that it might narrow the accuracy gap between linear incremental parsing and cubic time top-down parsing:

- We want to investigate the possibility of having joint simultaneous taggers for supertags and operators, such that each tagger is informed with the other tagger possible decision. This would enable the usage of the states as features for both taggers which can have a good effect on the taggers accuracy.

- The current implementation of the parser maintains only the best parsing decision at each state, but maintaining a limited number of possible states would enhance the parser's accuracy. However, this should be handled with an adequate graph search strategy such as A* search to keep the search space reasonable.

The techniques proposed in this chapter can be utilized in a different way to *linearize* any dependency graph. We can train a supertagger and an operator tagger to assign supertags and opertaor tags while having access to features from the dependency graph itself. Thus, we can use any dependency parser such as (Nivre, 2004; Shen and Joshi, 2005; Clark and Curran, 2007) to produce dependency structures for any available data. Then, the dependency-informed taggers are used to assign supertags and opertors which should represent a linearization of dependency structures. This indicates that we may use any dependency parser to construct our incremental dependency-based language model (IDLM).

In the next chapter, we will show how we make use of our incremental dependency-based language model (IDLM) to improve the translation quality of SMT.

Chapter 6

Dependency-based SMT

6.1 Introduction

Syntactically-enriched language models (cf. Section 3.6) constitute a promising component for SMT. These syntax-based language models, if integrated within an MT framework, can produce more grammatical translations by two means. First, they can enable constituency (cf. Section 3.1) by allowing constituent units of the translation to undergo long-range re-ordering while maintaining the grammaticality and the logical meaning of the units. Secondly, the subcategorization and dependency information can provide non-local, long-range relations such that it can enable long-range reordering while maintaining the grammatical structure of the translation output. However, to maintain a useful level of accuracy, existing parsers are non-incremental and must span a combinatorially growing space of possible structures as every input word is processed. This prohibits their incorporation into standard linear-time MT decoders. Moreover, most existing parsers deploy PCFG techniques which cannot handle non-constituent constructions commonly used in Phrase-based SMT systems.

In Chapter 5, we presented Incremental Dependency-based Language Model (IDLM) using incremental, linear-time dependency parser based on Combinatory Categorial Grammar (CCG). IDLM maintains a limited number of parse-states at each prefix of the sentence and so is very efficient for large-scale SMT systems. Since it is based on CCG, IDLM can handle non-constituent constructions (cf. Chapter 5) which are commonly found in Phrase-based SMT systems. In this chapter, we present a dependency-based SMT model which deploys IDLM and constructs the target language dependency structure incrementally as the translation proceeds step-by-step.

The remainder of this chapter is organized as follows. In Section 6.2 we present the general representation for incorporating IDLM into SMT systems. In Section 6.3 we review the related work. In Section 6.4 we discuss our choice for the baseline system in this chapter. In Section 6.5, we detail our approach. In Section 6.6, we introduce the experiments and the results. In Section 6.7, we introduce results analysis along with systems output examples. Finally, Section 6.8 concludes, and discusses future work.

6.2 Dependency-based Language Model for SMT

6.2.1 IDLM Representation for SMT

As it processes an input sentence left-to-right word-by-word, IDLM builds —for each prefix of the input sentence— a partial parse that is a subgraph of the partial parse that it builds for a longer prefix. The dependency graph is constructed incrementally, so at each step the constructed subgraph is never altered or revised in any later step. IDLM, as an incremental parser, is more appealing for large-scale applications as its time and space (worst-case) complexities are linear in input length. IDLM, an incremental and linear-time parser, constitutes a natural match for the word-by-word decoding and pruning schemes used within phrase-based SMT systems.

$$S_0 \xrightarrow[w_1,st_1]{o_1} S_1 \xrightarrow[w_2,st_2]{o_2} S_2 \cdots\cdots\cdots S_i \xrightarrow[w_i,st_i]{o_i} S_{i+1} \cdots\cdots\cdots S_n \tag{6.1}$$

For incremental parsing in the monolingual case, as we discussed in Chapter 5, the IDLM syntactic process is represented by a sequence of transitions between adjacent syntactic states. The syntactic representation is built step-by-step from left-to-right while traversing the input string as shown in (6.1). The syntactic state is supposed to summarize all the syntactic information about fragments that have already been processed so far. The parser produces fully connected intermediate structures while moving from one word to the next.

For MT, the same process applies except that the target words/phrases are the candidate translations of the source words/phrases. Each target word/phrase represents a structure or sub-graph composed of the lexical words, with associated supertag and operator sequences. As shown in (6.2), each source phrase can be translated to a target phrase structure. In this structure, each word w_i is associated with a lexical syntactic/semantic descriptor st_i and a possible parsing action (operator) o_i that may take place with this word/phrase-supertag pair. These sub-graphs along with their probabilities represent our phrase table augmented with incremental dependency parsing support.

$$s_i ... s_n \longrightarrow [w_i, st_i, o_i] ... [w_n, st_n, o_n] \tag{6.2}$$

6.2.2 Linear-time, Incremental Parsing Decoder

As it processes the source sentence left-to-right, word-by-word, the decoder expands each translation hypothesis with the possible translations for this source word/phrase. The translations are associated with possible supertag and operator sequences as discussed above. The decoder specifies and maintains a parse-state for each decoding hypothesis state. Each parse-state is represented by a composite CCG category which is the result of applying the combinatory operator sequence to the preceding parse-state and the current phrase supertag sequence. The parse-state CCG composite category specifies a functor and its arguments are the expected categories while expanding the current hypothesis.

Based on (6.1), each state S_i is determined exactly by the previous state S_{i-1}, and a choice of a supertag st_i and an operator o_i. Therefore, the probability $P(W, S)$ of a word sequence W and associated final parse-state sequence S, which represents a possible derivation, can be described as in Eqn (6.3). The probability $P(W, S)$ represents the product of the state production probabilities at each parse-state:

$$P(W, S) = \prod_{i=1}^{n} P(w_i|W_{i-1}S_{i-1}).P(st_i|W_i).P(o_i|W_i, S_{i-1}, ST_i) \tag{6.3}$$

In Eqn (6.3):

- $P(w_i|W_{i-1}S_{i-1})$ is the probability of w_i given the previous sequence of words W_{i-1} and the previous sequence of states S_{i-1}.
- $P(st_i|W_i)$: is the lexical descriptor (supertag st_i) probability given the word sequence W_i up to the current position. This is represented by a sequence tagger (supertagger) in our CCG incremental parser.
- $P(o_i|W_i, S_{i-1}, ST_i)$ represents the parsing action (operator o_i) probability given the previous words, supertags and state sequences up to the current position. This is represented by a sequence operator tagger in our CCG incremental parser.

Crucially, given a sentence and its state sequence, the dependency structure can be retrieved unambiguously. At each state the partial dependency structure can be represented as a directed graph with nodes representing words and arcs representing dependency relations.

Although the above outlined framework matches the nature of Phrase-based SMT systems, further attention should be paid to two issues. First, an efficient representation for the phrase tables is needed to avoid an explosion of the phrase space. Since each phrase is associated with a number of supertag sequences and a number of operator sequences, this could simply lead to very large phrase tables with sparse probabilities which in turn complicates the decoding process. Second, although IDLM maintains a single state for each hypothesis, the search space will be much larger than the case without IDLM and this needs efficient handling to avoid a further explosion of the search space. In the next section we will review the related work.

6.3 Related Work

In Section 2.5, we discussed previous approaches for incorporating syntax into Phrase-based SMT and we highlighted their limitations. In Section 3.6, we reviewed various syntax-based language models and highlighted their limitations as well. In this section we review a very recent approach using a dependency-based language model for SMT and we contrast this approach and our own.

(Shen et al., 2008) introduced an interesting approach for incorporating a dependency-based language model into SMT. They proposed to extract String-to-Dependency trees from the parallel corpus. As the dependency trees are not constituents by nature, they are able to handle non-constitute phrases as well. While this work shares the same target as ours, namely incorporating dependency parsing into SMT, there remain three major differences. Firstly, (Shen et al., 2008) resorted to some heuristics to extract the String-to-Dependency trees while our approach deploys a more formalized grammatical theory. Secondly, their decoder works bottom-up and uses a chart parser with limited language model capability (3-gram), while we use the more efficient linear decoder commonly used in Phrase-based SMT. Thirdly, (Shen et al., 2008) deploys the dependency language model to augment the lexical language model probability between two head words similar to (Xu et al., 2002) and never seek a full dependency graph. In contrast, our approach integrates a fully incremental parsing capability that produces the dependency structures while decoding and thus provides better guidance for the decoder to construct more grammatical output. To the best of our knowledge, our approach is the first to incorporate fully incremental dependency parsing capabilities into SMT with linear time and space decoding.

6.4 Difficulties of incorporating IDLM into Phrase-based SMT

In Chapter 4, we extended the Phrase-based SMT system with supertagged translation and language models by adding a number of log-linear features to the model such that we had a total of eleven log-linear features in our model. We also show in Section 4.6 that we have resorted to some *ad hoc* methods to be able to tune this relatively large number of system parameters. As we discussed in Section 2.6.1, the limited capability of MERT estimation represents a bottleneck to further serious development of features-rich SMT systems, as has just been highlighted in (Chiang et al., 2008), who proposed a new method to estimate up to 56 parameters.

For integrating IDLM, we definitely need more features than the supertagged Phrase-based model in which we added five features to support just the supertags. We need to represent supertags, operators, states and various conditional probabilities between them and other features in the system. In the light of the above limitations, we think that integrating IDLM into SMT needs a more sophisticated system that can support many features without such a limitation in the estimation process. Fortunately, discriminative direct translation models (DTM2) (Ittycheriah and Roukos, 2007) allows the use of millions of features in a more formalized probabilistic framework with optimal estimation techniques. Based on these factors, we opted for DTM2 as the framework for integrating our IDLM into SMT. We think that DTM2 is a more formalized framework and will allow the exploration of a wide variety of possible features in a unified modeling framework.

6.5 Dependency-based Direct Translation Model (DDTM)

6.5.1 Model Overview

We reviewed Direct Translation Models (DTM) in detail in Section 2.6. DTM models the *a posteriori* conditional distribution $P(T|S)$ instead of $P(S|T)$ as in the source channel approach. DTM has three components: a prior conditional distribution $P_0(T|S)$, a number of feature functions that capture the translation and language model effects in a unified framework and finally weights of the features that can be estimated by MaxEnt (Berger et al., 1996). (Ittycheriah and Roukos, 2007) introduced DTM2 to handle Phrase–based SMT using a minimum number of phrases with no overlap and finally training the whole set of millions of system parameters using MaxEnt.

We extended DTM2 to support our incremental dependency-based language model (IDLM) introduced in Chapter 5. The target-side sentences are augmented with supertag, operator and state sequences. DTM2 was extended by incorporating the model introduced in Eqn. (6.3) as a set of MaxEnt features, as we will discuss in detail later.

This representation turns the complicated problem of MT with incremental parsing into a sequential classification problem in which the classifier deploys various features from the source sentence and the candidate target translations to specify a sequence of decisions that finally results in an output target string along with its associated dependency graph. The classification decisions are performed in sequence step-by-step while traversing the input string to provide decisions on possible words, supertags, operators and states. A beam search decoder simultaneously decides which sequence is the most probable.

$$T^* = \arg\max_T P(T|S) = 1/Z \quad exp \sum_i \lambda_i \phi_i(S,T) \tag{6.4}$$

As shown in Equation (6.4), Phrase–based SMT is represented as a classification problem with arbitrary features defined over the source and the target. More specifically, the reordering and prior phrase probabilities are represented as shown in equation (6.5).

$$P(T|S) = P_0(T,J|S)/Z \quad exp \sum_i \lambda_i \phi_i(T,J,S) \tag{6.5}$$

Here P_0 is the prior distribution for the phrase probability which is usually the phrase normalized counts used in any conventional Phrase–based SMT system. J is the skip reordering factor for this phrase pair which represents the jump from the previous source word.

6.5.2 DDTM Features

In our DDTM, we have implemented many features along with the baseline DTM2 features that we have discussed in Section 2.6. We have extended DTM2 with a number of features to represent the incremental dependency-based language model as listed here:

- Supertag-Word features: these features examine the target phrase words with their associated supertags.
- Supertag sequence features: these features encode *n*-gram supertags (equivalent to the *n*-gram supertags Language Model).
- Supertag-Operator features: these features encode supertags and their associated operators.
- Supertag-State features: these features encode states and supertags co-occurrence.
- State sequence features: these features encode *n*-gram states features and are equivalent to an *n*-gram states Language Model.
- Word-State sequence features: these features encode words and states co-occurrence.

The features described above encode all the probabilistic components in Eqn. (6.3) along with some more empirically intuitive features.

6.5.3 DDTM Decoder

The decoder adopted in the baseline DTM2 (Ittycheriah and Roukos, 2007) is a beam search decoder similar to decoders used in standard phrase-based log-linear systems such as (Tillmann and Ney, 2003) and (Koehn, 2004a). The main difference between the DTM2 decoder and the standard Phrase–based SMT decoders is that DTM2 deploys Maximum Entropy probabilistic models to obtain the translation costs and various feature costs by deploying the features described above in a discriminative MaxEnt fashion.

In order to support incremental dependency parsing, the decoder has been extended in three main ways: firstly, by constructing the syntactic states during decoding; secondly, by extending the hypothesis structures to incorporate the syntax states and the partial dependency derivations; and thirdly, by modifying the pruning strategy to handle the large search space.

At decoding time, each hypothesis state is associated with a parse-state which is constructed while decoding using the Parse State Realizer (identical to the parse-states and the Realizer introduced in Section 5.6.2). The Parse-State Realizer is a deterministic module that deploys the previous state, the sequences of supertags and CCG incremental operators to realize the parse-states as well as the intermediate dependency graphs between words.

Figure 6.1 shows the DDTM decoder while decoding a sentence with the English translation “Attacks rocked Riyadh”. Each hypothesis is associated with a parse-state Si and a partial dependency graph (shown for some states only). Moreover, each transition is associated with an operator O that combines the previous state and the current supertag ST to construct the next state Si. The decoder starts from a null state $S1$ and then proceeds with a possible expansion with the word “attacks”, supertag NP and operator NOP to produce the next hypothesis with state $S2$ and category NP. Further expansion for that path with the verb “rocked”, supertag ‘$(S\backslash NP)/NP$ and operator $TRFC$ will produce the state $S5$ with category S/NP. The partial dependency graph for state $S5$ is shown above the state where a dependency relation between the two words is established. Furthermore, another expansion with the word “Riyadh”, supertag NP and operator FA produces state $S7$ with category S and a completed dependency graph as shown above the state. Another path which spans the states $S1$, $S3$, $S6$ and $S8$ ends with a state category

S/NP and a partial dependency graph as shown under state $S8$ where the dependency graph is still missing its object.

Figure 6.2 shows partial decoding graph for a longer sentence, with complete paths. Each hypothesis is associated with a parse-state.

The addition of parse-states may result in very large search space due to the fact that the same phrase/word may have many possible supertags and many possible operators. Moreover, the same word sequences may have many parse-state sequences and, therefore, many hypotheses that represent the same word sequence. The search space is definitely larger than the baseline search space. We adopt the following three pruning heuristics to limit the search space.

Grammatical Pruning

Any hypothesis which does not constitute a valid parse-state is discarded, i.e. if the previous parse-state and the current supertag sequence cannot construct a valid state using the associated operator sequence, then the expansion is discarded. Therefore, this pruning strategy maintains only fully connected graphs and discards any partially connected graphs that might result during the decoding process.

As shown in Figure 6.1, the expansion from state $S1$ to state $S4$, with the dotted line, is pruned and not expanded further because the proposed expansion is the verb "attacks", supertag $(S\backslash NP)/NP$ and operator $TRFC$. Since the previous state is NULL, it cannot be combined with the verb using the $TRFC$ operator. This would produce an undefined state and thus the hypothesis is discarded.

Supertags and Operators Threshold

We limit the supertag and operator variants per target phrase to a predefined number of alternatives. We tuned these thresholds using the MT03 DevSet. The supertags limit was set to four alternatives while the operators limit was set to three alternatives. We tuned these thresholds for the best accuracy while maintaining a manageable search space.

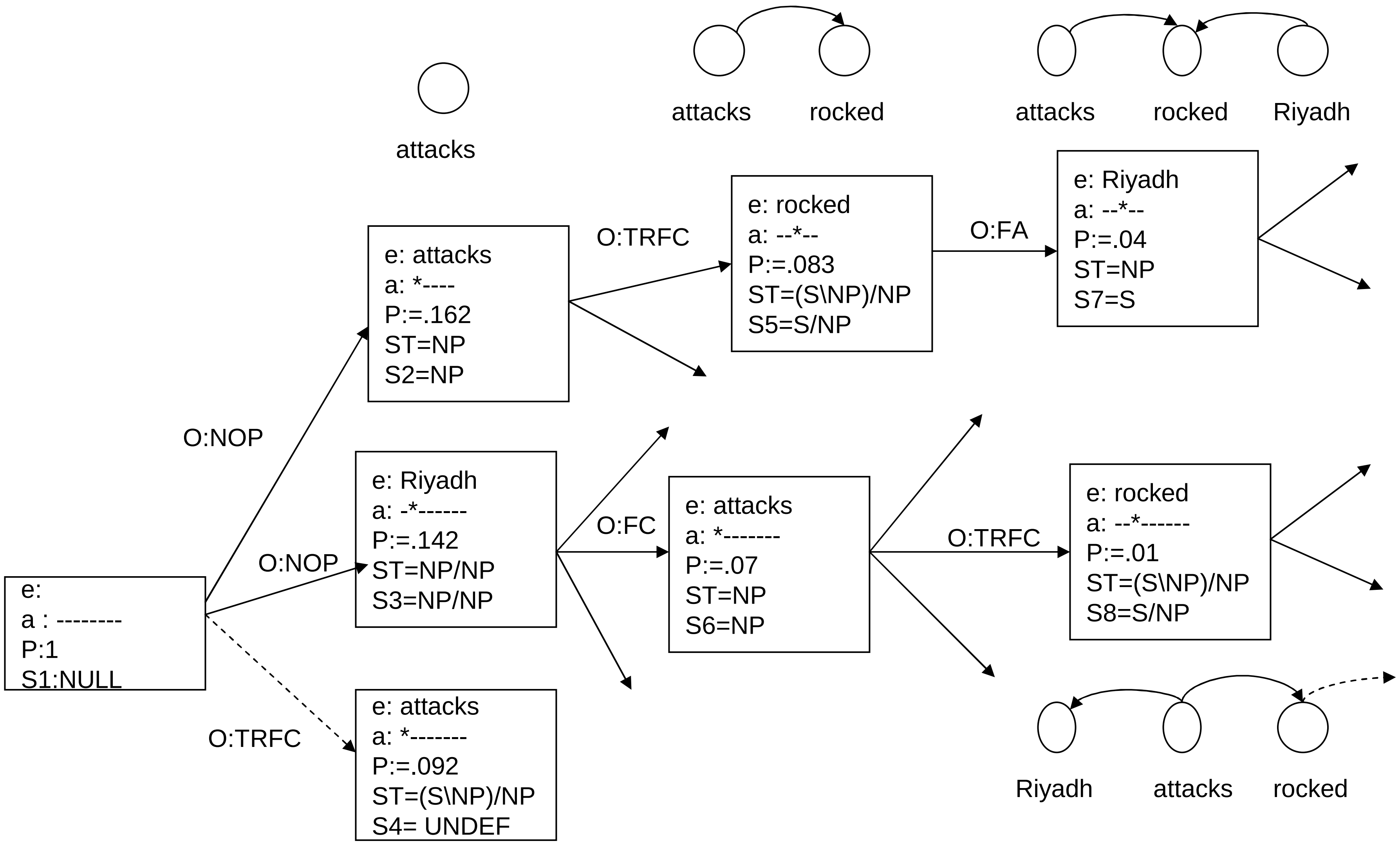


Figure 6.1: DDTM Decoder: each hypothesis has a parse state and a partial dependency structure

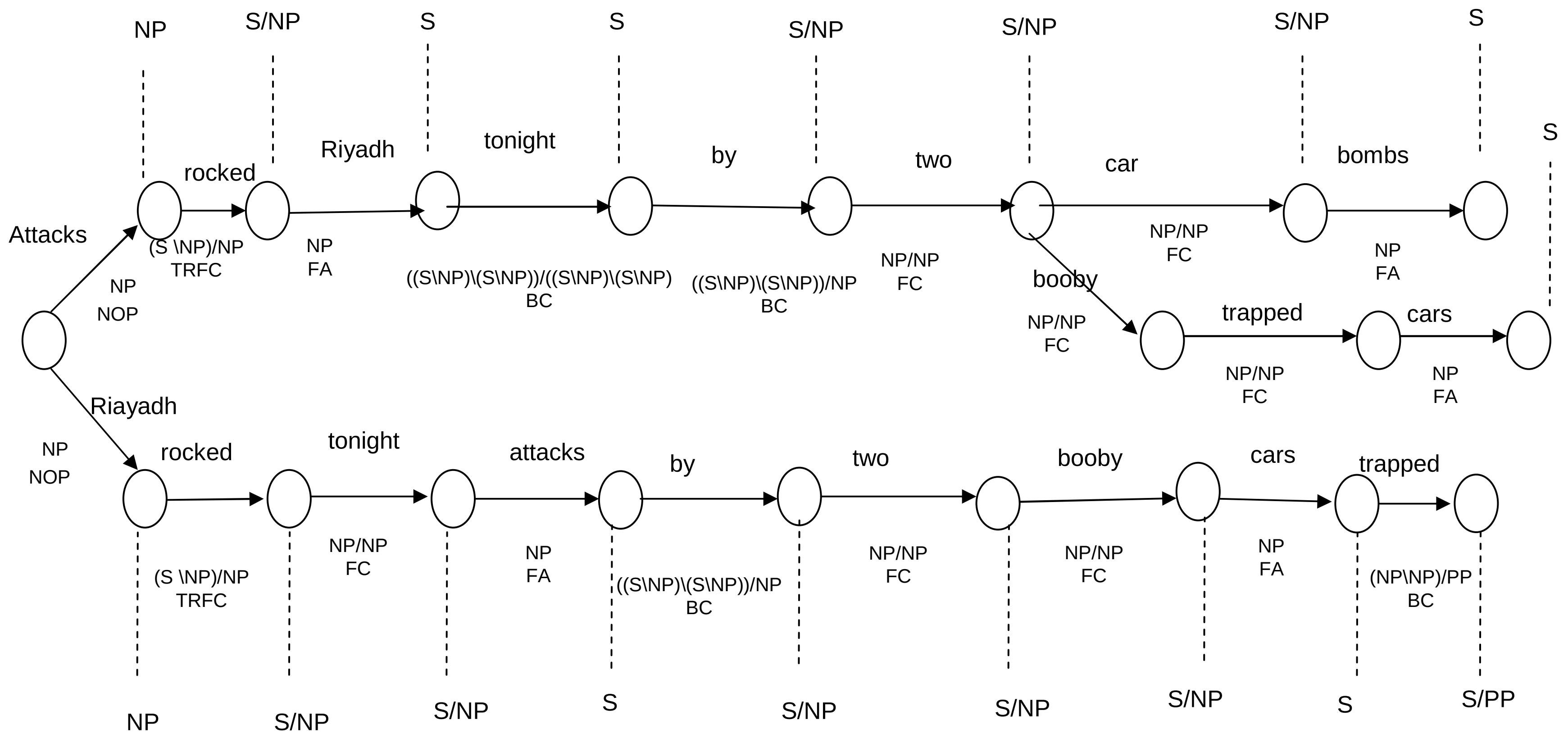


Figure 6.2: Partial decoding graph for a sentence: We show hypothesis paths along with supertags and operators associated with each word and the parse-state category associated with each decoding state.

As shown in Figure 6.1, each word can have many alternatives with different supertags. In this example the word “attacks” has two forms, namely a noun and a verb, with different supertags and operators. The proposed thresholds limit the possible alternatives to a reasonable number.

Merging Hypotheses

Standard Phrase–based SMT decoders (cf. Section 2.4) merge translation hypotheses if they cover the same source words and share the same *n*-gram language model history. Similarly, DDTM decoder merges translation hypotheses if they cover the same source words, share the same *n*-gram language model history and share the same parse-state history. This helps in reducing the search space by merging paths that will not constitute a part of the best path.

6.6 Experiments

We conducted an extensive set of experiments to examine the proposed approach and its features. In this set of experiments we used the UN parallel corpus and LDC news corpus together containing 3.7M parallel sentences. The lexical 5-gram LM was trained on the English Gigaword Corpus. Our baseline system is the DTM2 model described in (Ittycheriah and Roukos, 2007) and outlined in Section 2.6.

In order to train our DDTM model, we used the incremental parser introduced in Chapter 5 to parse the target side of the parallel training data. Each sentence is associated with supertag, operator and parse-state sequences. We then trained various models with different features.

Although we used our incremental parser described in Chapter 5, any dependency parser, whether incremental or not, such as (Nivre, 2004; Shen and Joshi, 2005; Clark and Curran, 2007) can be used to process the training data. As we highlighted in Section 5.8, using our approach any dependency structure can be linearized into incremental form with

CCG grammar. Indeed, we tried to use the ‘C&C’ dependency parser[1] (Clark and Curran, 2007) as its accuracy is higher than our incremental version; unfortunately more than 30% of the training data sentences cannot obtain a parse using the ’C&C’ parser at all.

6.6.1 Results

We compared two baseline systems with our DDTM using the features listed above. The first baseline is IBM Phrase–based SMT system (Al-Onaizan and Papineni, 2006) while the second is the DTM2 system. Table 6.1 shows which features are used in which system.

Features/System	DTM2	D-SW	D-SLM	D-SO	D-OLM	D-SS	D-WS	D-SLM	DDTM
Baseline features	X	X	X	X	X	X	X	X	X
Supertag-Word		X	X	X	X	X	X	X	X
Supertag ngram			X	X	X	X	X		X
Supertag-Operator				X	X	X	X	X	X
Operator *n*-gram					X				
Supertag-State							X	X	X
State-Word							X		
State *n*-gram								X	X

Table 6.1: DDTM systems with associated features

Generally we examined all features to realize their effect on the system. The systems examined are:

- IBM-PB: IBM Phrase–based SMT baseline system.
- DTM2: the baseline Direct Translation model system.
- D-SW: examines Supertag-Word features.
- D-SLM: examines Supertag-Word features and supertags *n*-gram features.
- D-SO: examines Supertag-Operator features.
- D-OLM: examines operator *n*-gram features.
- D-SS : examines supertags and states features with parse-state construction.

[1]http://svn.ask.it.usyd.edu.au/trac/candc

- D-WS : examines words and states features with parse-state construction.
- D-SLM: examines *n*-gram states features with parse-state construction.
- DDTM: fully fledged system with all features that proved useful above.

System	**BLEU Score on MT05**
IBM-PB	50.16
DTM2-Baseline	52.24
D-SW	52.28
D-SLM	52.29
D-SO	52.01
D-OLM	51.87
D-SS	52.39
D-WS	52.03
D-SLM	52.53
DDTM	52.61

Table 6.2: DDTM Results with various features.

As shown in Table 6.2, the DTM baseline system demonstrates a very high BLEU score. It is worth mentioning that the baseline system is already top-ranked in two recent major MT evaluations. Among the features we tried, supertags and *n*-gram supertags systems (D-SW and D-SLM systems) give slight yet statistically insignificant improvements. On the other hand, the states *n*-gram sequence features (D-SS and DDTM systems) give a small yet statistically significant improvements. The operators *n*-gram features (D-OLM system) show a remarkable degradation of the system. This shows that the operators sequence, on its own, is not an important factor to guide the structure without the corresponding supertags and states. Similarly, the states-word features (D-SW system) show a degradation. This may be due to the fact that the states-words interaction is very sparse and could not be estimated with good evidence.

We might expect that using an MT evaluation metric such as (Owczarzak et al., 2007), that takes into account the matching of the dependency relations between the system translation and the references, would give a better result. However, we tried this evaluation metric for some of the systems reported above and we found that the relative differences between system scores with this metric are similar to the differences provided by BLEU. In any case, we think that evaluating MT systems that incorporate dependency information using MT evaluation metrics that measure dependency relation matching is as unfair as evaluating *n*-gram-based systems using the BLEU score (Callison-Burch et al., 2006). As a matter of fact, we think that our proposed model would have a better chance using human evaluation; in the last year IWSLT-07 evaluation, our supertags-based Arabic–English system described in (Hassan et al., 2007a) was judged to be ranked first by some margin in the human evaluation, despite being ranked 5th in the automatic evaluation with 2 BLEU points less than the first system in the automatic evaluation (Fordyce, 2007).

6.7 Results Analysis

Although the BLEU score did not show a remarkable improvement by the dependency-based system over the baseline sysem, human inspection of the data gives us important insight into the pros and cons of the dependency-based model. The examples here show a consistent behaviour of the baseline and the DDTM systems which can be observed in many examples throughout the test set. We only highlight some of the examples for illustration purposes.

The example in Figure 6.3 shows how DDTM manages to insert verb “reported” instead of the phrase “according to”. Usually DDTM prefers to deploy verbs since they have complex and more detailed syntactic structures which give better and more likely state sequences. Furthermore the example shows how DDTM avoids longer noun phrases and instead uses some prepositions in between; the baseline opted for “cali cartel leader”, while DDTM preferred “the leader of cali cartel”. Again, this may be due to the fact that prepositions have a complex syntactic description that may give rise to a more likely state sequence.

Figure 6.4 shows two examples where DDTM provides better and more concise syntactic structure. As we can see, there is not much agreement between the reference and

Source: بوغتا ٤-١٢ (ا ف ب) - ذكر مراسل وكالة فرانس برس ان زعيم كارتل كالي (جنوب غرب) جيلبرتو رودريغس اوريهول ، احد اكبر مهربي المخدرات في العالم ، سلم مساء الجمعة الي الولايات المتحدة .

Reference: *Bogota 12-4 (AFP) - An Agence France-Presse correspondent reported th at Cali cartel boss (south-west) Gilberto Rodriguez Orejuela, one of the biggest drug traffickers in the world, was handed over to the United States on Friday e vening.*

Baseline: *Bogota 4-12 (afp) - according to an Agence France Presse correspondent that cali cartel leader (southwest) , gilberto rodriguez orejuela , one of the biggest drug traffickers in the world , surrendered friday night to the united states .*

DDTM: *Bogota 4-12 (afp) - An Agence France Presse correspondent reported that the leader of the cali cartel (southwest) Gilberto Rodriguez Orejuela , one of the biggest drug traffickers in the world , handed over friday night to the United States .*

Figure 6.3: DDTM opts for inserting verbs and breaking long noun phrases with prepositions.

Source: وخضع بعد ذلك لفحوصات اجراها احد اطباء الشرطة

Reference: *He then underwent medical examinations by a police doctor .*

Baseline: *He was subjected after that tests conducted by doctors of the police .*

DDTM: *Then he underwent tests conducted by doctors of the police .*

Source: وقد هز الرياض مساء اليوم هجومان بسيارتين مفخختين

Reference: *Riyadh was rocked tonight by two car bomb attacks..*

Baseline: *Riyadh rocked today night attacks by two booby - trapped cars.*

DDTM: *Attacks rocked Riyadh today evening in two car bombs.*

Figure 6.4: DDTM provides better syntatctic structure with more concise translation.

the proposed translation. However, longer translations enhance the possibility of picking more common *n*-gram matches via the BLEU score and so increases the chance of better scores. This is not in favour with the more concise DDTM output.

The example shown in Figure 6.5 shows a better translation by the baseline. The baseline lexical language model made a better job here as it is not a likely *n-gram* that "prime minister meets the capital", whereas DDTM opted for a different syntactic structure. We think such problems can be solved with a light lexicalization of the verbs' predicate-argument structures in our framework. We could use features that encode the lexicalization of the subject-object frames of the verbs such that the features would prefer

Source: وسيلتقي رئِيس الوزرَاء رجب طيب اردوجان قبل ان يغادر العاصمة التركية مساء

Reference: *He will meet Prime Minister Recep Tayyip Erdogan before leaving the Turkish capital in the evening.*

Baseline: *He will meet prime minister Recep Tayyip Erdogan before leaving the turkish capital in the evening .*

DDTM: *Prime minister Recep Tayyip Erdogan will meet before he leaves the turkish capital in the evening .*

Figure 6.5: Example: Long range reordering and the need for lexicalization.

the bilexical relation "meet-minister" over the bilexical relation "meet-capital". Similarly, the bilexical relation "organization-announced" should be preferred over "organization-said".

Source: وتنهي اليوم الثلاثَاء المهلة الدستورية لـلرئِيس الموقت روحي فتوح بحسب ابو صلاح .

Reference: *According to Abu Salah, today, Tuesday, is when the constitutional period of the Interim President Rouhi Fattouh expires.*

Baseline: *Ends today , Tuesday , the constitutional deadline for the interim president Rouhi Fattouh and according to Abu Salah .*

DDTM: *Today , Tuesday , the constitutional deadline to end the interim president Rouhi Fattouh , according to Abu Salah .*

Figure 6.6: Better long-range reordering

The example shown in Figure 6.6 shows how DDTM manages to handle syntactic-based long-range reordering (9 positions here), which resulted in better syntactic structure and better translation in general.

6.8 Conclusion

In this chapter, we have presented a novel model of dependency phrase-based SMT which integrates fully incremental dependency parsing into the translation model while retaining the linear decoding assumed in conventional Phrase–based SMT systems. To the best of our knowledge, this model is the first model to integrate dependency parsing into Phrase–based SMT systems with linear decoding. Our model is based on the novel IDLM which deploys dependency parsing to provide incremental parser information in the translation system. Moreover, our proposed approach integrates the capability of full dependency parsing in SMT systems with a very attractive computational cost since it still deploys the linear decoders widely used in Phrase–based SMT systems.

We carried out extensive experiments on a very large training set and a standard widely used test set for Arabic–English translation. While we did not observe a huge improvement over the already top-ranked baseline system, we believe that the proposed approach can provide better translation quality especially in human evaluations.

As we show in the last section, incremental dependency parsing in the form of our proposed dependency language model can make better syntactic structures available to the MT output. Syntactic-informed long-range reordering and constituency enablement are also introduced such that constituent units can undergo long-range reordering while maintaining grammaticality. All of these aspects can help to produce better, more grammatical MT output.

Our DDTM system could be further expanded in many dimensions. As we noted while analyzing the system output, some light lexicalization features could be of benefit to the system. Furthermore, we could examine the possibility of using the dependency information encoded in the CCG categories as features in the system.

Finally, the approach introduced here can be extended to include logical semantic relations as well using the CCG syntactic/semantic interface, which would be a further step on the right direction of producing better MT output.

Chapter 7

Conclusions

7.1 Contribution of the Thesis

In this thesis, we extended Phrase-based SMT with lexical syntactic descriptions — supertags— that localize global syntactic information on the word level. Supertags can, therefore, be assigned to every word in a phrase without introducing syntactic redundant ambiguity. We introduced two different levels of syntactic support namely:

- Incorporating supertagged translation model and supertagged *n*-gram language model into Phrase-based SMT.
- Incorporating incremental dependency-based language model into DTM2.

Both approaches proved to be useful for enhancing the translation quality and providing more grammatical translations.

We presented a novel model of Phrase-based SMT which integrates supertags into the target side of the translation model and the target language model. We carried out extensive experiments on small and very large training and test sets for Arabic–English and German–English translation. While using LTAG supertags gives the best improvement over a state-of-the-art Phrase-based SMT system for the smaller data set, using CCG supertags works best on the large training set. The experiments on very large training data provided evidence that an ever increasing amount of data will not bridge the performance gap with a system that incorporates syntactic information.

We provided an in-depth manual analysis of the system performance. We showed that a very wide range of improvements were brought about by the use of a supertags-based system, including improved reordering, overcoming the tendency of SMT systems to omit verbs, improved verbal constructions, proper handling of negation, and better syntactic modeling in general. We noted that in a recent open evaluation, the output from our Arabic–English supertagged system (Hassan et al., 2007a) was ranked first by human evaluators reflecting the fact that lexical syntax can produce more grammatical and fluent translations despite the fact that today's automatic evaluation metrics cannot capture such effects.

The encouraging results of our proposed supertagged Phrase-based SMT approach provided a momentum to investigate further opportunities for improvements using lexical syntax. We introduced our Incremental Dependency-based Language Model (IDLM) based on wide-coverage CCG incremental parsing. The proposed dependency-based LM has very interesting characteristics that facilitates its integration into Phrase-based SMT systems. First, the language model parser is deterministic in that it maintains a limited number of parsing decisions at each state which makes it very efficient for integration into large-scale Phrase-based SMT systems. Second, it is incremental in Markovian fashion similar to Phrase-based SMT decoders. Third, it can naturally handle non-constituent constructions, being based on CCG. Fourth, the parser always seeks fully connected structures, not just using syntactic information to augment LM probabilities. At the same time, the parser can handle non-connected structures as well. Fifth, the parser supports long-range dependencies and a number of interesting syntactic phenomena in a fully incremental left-to-right fashion.

Furthermore, we developed an incremental version of the CCGbank that can be used to train such an incremental parser. The techniques deployed in the conversion can be used to linearize any dependency structure so that it can be used in language modeling.

Finally, we have incorporated our IDLM into the direct translation model (DTM2) while retaining the linear decoding assumed in conventional Phrase–based SMT systems. To the best of our knowledge, this model is the first to integrate dependency parsing directly into Phrase–based SMT systems with linear decoding. Our model is based on the novel IDLM which deploys dependency parsing to provide incremental parsing information to the translation system. Moreover, our proposed approach integrates the capability of full dependency parsing in SMT systems with a very attractive computational cost since it still deploys the linear decoders widely used in Phrase–based SMT systems.

We carried out extensive experiments on a very large training set and a standard widely used test set for Arabic–English translation. While we did not observe a huge improvement over the already top-ranked baseline system, we believe that the proposed approach can provide better translation quality especially in human-based evaluations such as HTER (cf. Section 2.7).

We carried out an extensive analysis of the system output and demonstrated that the incremental dependency parsing in the form of our proposed dependency language model can make better syntactic structures available to the MT output. Syntactically-informed long-range reordering and constituency enablement is also introduced such that constituent units can undergo long-range reordering while maintaining grammaticality. All of these aspects can help produce better and more grammatical MT output.

Recalling our research questions that we have introduced in Chapter 1:

RQ1: What is the grammatical representation that can fit with Phrase-based SMT while not introducing redundantly ambiguous syntactic structures?

We found that Phrase-based SMT can be extended with lexicon-driven approaches to linguistic syntax, namely Lexicalized Tree-Adjoining Grammar (Joshi and Schabes, 1991) and Combinatory Categorial Grammar (Steedman, 2000). These lexical syntactic descriptions localize global syntactic information on the word level; therefore, they can be assigned to every word in a phrase without introducing redundant ambiguity.

RQ2: How to incorporate lexical syntax descriptions into Phrase-based SMT while maintaining its advantages and does it help in providing better and more grammatical translation?

We presented a novel model of Phrase-based SMT which integrates supertags into the target side of the translation model and the target language model. Our proposed approach provided significant improvements over state-of-the-art systems for two different language pairs.

RQ3: Does Phrase-based SMT need more syntactic knowledge or our supertagged approach is sufficient for providing syntactic structures to enable more grammatical translations and better reordering?

We Showed that MT needs a more sophisticated mechanism that can support long-range dependencies, construct full parse structures, work in an incremental manner and be computationally efficient.

RQ4: Can lexical syntax provide more syntactic knowledge for Phrase-based SMT through incremental dependency parsing capabilities that match the nature of Phrase-based SMT?

We introduced our Incremental Dependency-based Language Model (IDLM) based on wide-coverage CCG incremental parsing. Our IDLM is deterministic, incremental in Markovian fashion and naturally handle non-constituent constructions, being based on CCG.

RQ5: Is it possible to incorporate full incremental dependency parsing into SMT while maintaining SMT scalability and computationally efficient linear decoding?

we have incorporated our IDLM into the direct translation model (DTM2) with linear decoder. Our approach provided good improvements over a top-ranked baseline system.

We summarize here the major contributions of this thesis :

- We introduced a novel model of supertagged Phrase-based SMT which integrates supertags into the target language model and the target side of the translation

- We introduced a novel dependency-based LM which is deterministic in that it maintains a limited number of parsing decisions at each state which. Furthermore, it is incremental in Markovian fashion similar to Phrase-based SMT decoders and it can naturally handle non-constituent constructions, being based on CCG.

- We introduced an extension to direct translation models that integrates incremental dependency parsing while retaining the linear decoding assumed in conventional Phrase–based SMT systems.

7.2 Future Research Avenues

This thesis provides many directions for future research opportunities. The incremental dependency-based parser offers many opportunities for enhancements. The parser dependency accuracy could be enhanced using joint simultaneous taggers for supertags and operators, such that each tagger is informed with the other tagger possible decisions. This would enable the usage of the states as features for both taggers which should have a good effect on the taggers accuracy. Another possibility of enhancing the parser is adding an adequate graph search strategy such as A* search, so that the parser is able to keep a reasonable number of states instead of single state, as is the case now.

The dependency linearization techniques that we introduced to convert the CCGbank into incremental form need a more thorough study from the graph representation point of view. We may want to know which kind of dependency graphs could be linearized, what is the limitation of this linearization and whether there are more formalized ways to perform such linearization.

Our DDTM system could be further expanded in many dimensions. For example, some light lexicalization features could be of benefit to the system. Using the dependency information encoded in the CCG categories as explicit features may help as well.

A possible future direction is to include supertags and dependency information from the source side as well. This would help to obtain target structures in correspondence with source structures. However, currently there are only a few languages for which supertag-sets and supertaggers exist, which limits such possible extensions. We hope that the work presented in this thesis may encourage other researchers to investigate the possibility of bootstrapping supertags for more languages.

7.3 Closing Remark

We believe that this thesis puts the first corner stone into a fully integrated lexicalized, syntactic and semantic framework. In this thesis, we have just scratched the surface where lexicalized syntactic translation is concerned; however, we believe that the same framework can be extended to include logical semantic relations as well using the CCG syntactic/semantic interface, which would be a further step in the right direction to produce better MT output.

Bibliography

Abney, S. (1991). Parsing By Chunks. In Berwick, R., Abney, S., and Tenny, C., editors, *Principle-Based Parsing*, pages 257–278. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Ajdukiewicz, K. (1935). Die syntaktische konnexitatn. *Polish Logic*, **1**(27):207–231.

Al-Onaizan, Y. and Papineni, K. (2006). Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics: ACL-44*, pages 529–536, Sydney, Australia.

Alshawi, H., Bangalore, S., and Douglas, S. (2000). Translation Models as Collections of Finite State Head Transducers. *Computational Linguistics*, **26**:45–60.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan.

Bangalore, S. (2000). A Lightweight Dependency Analyzer for Partial Parsing. *Natural Language Engineering*, **6**(2):113–138.

Bangalore, S., Emami, A., and Haffner, P. (2005). Factoring global inference by enriching local representations. *AT&T Research Labs*, Technical Report.

Bangalore, S. and Joshi, A. (1999). Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, **25**(2):237–265.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Languages*, **29**:47–58.

Bayraktar, M., Say, B., and Akman, V. (1998). An Analysis of English Punctuation: The Special Case of Comma. *International Journal of Corpus Linguistics*, **3**:33–58.

Berger, A., Della-Pietra, V., and Della-Pietra, S. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, **22**(1):39–71.

Birch, A., Osborne, M., and Koehn, P. (2007). CCG Supertags in Factored Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation, ACL-07*, pages 9–16, Prague, Czech Republic.

Brown, P., Cocke, J., Della-Pietra, S., Della-Pietra, V., Jelinek, F., Mercer, R., and Roossin, P. (1988). A statistical approach to language translation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, pages 71–76, Budapest, Hungary.

Brown, P., Cocke, J., Della-Pietra, S., Jelinek, F., Della-Pietra, V., Lafferty, J., Mercer, R., and Roossin, P. (1990). A Statistical Approach to Machine Translation. *Computational Linguistics*, **16**(2):79–85.

Brown, P., Della-Pietra, V., Della-Pietra, S., and Mercer, R. (1993). The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, **19**(2):263–311.

Callison-Burch, C., Osborne, M., and Koehn, P. (2006). Re-evaluation the role of bleu in machine translation research. In *Proceedings of 11st Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 249–256, Trento, Italy.

Charniak, E. (2000). A Maximum-Entropy-Inspired Parser. In *1st Annual Meeting of the North American Association for Computational Linguistics, Proceedings (NAACL 2000)*, pages 132–139, Seattle, WA.

Charniak, E. (2001). Immediate-head parsing for language models. In *39th Meeting of the Association for Computational Linguistics - (ACL'01)*, pages 124–131, Toulouse, France.

Charniak, E., Knight, K., and Yamada, K. (2003). Syntax-based language models for machine translation. In *Proceedings of the 9th Machine Translation Summit*, pages 40–46, New Orleans, LA.

Chelba, C. (2000). *Exploiting Syntactic Structure for Natural Language Modeling*. PhD thesis, Johns Hopkins University, Baltimore, MD.

Chen, J., Bangalore, S., and Vijay-Shanker, K. (2006). Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering*, **12**(3):251–299.

Chiang, D. (2005). A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL05)*, pages 263–270, Ann Arbor, MI.

Chiang, D., Marton, Y., and Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Honolulu, HI.

Clark, S. and Curran, J. (2004). The Importance of Supertagging for Wide-Coverage CCG Parsing. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, pages 282–288, Geneva, Switzerland.

Clark, S. and Curran, J. (2007). Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, **33**(1):439–552.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

Collins, M. and Roark, B. (2004). Incremental Parsing with the Perceptron Algorithm. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 111–118, Barcelona, Spain.

Collins, M., Roark, B., and Saraclar, M. (2005). Discriminative syntactic language modeling for speech recognition. In *43rd Annual Meeting on Association for Computational Linguistics - (ACL'05)*, pages 507–514, Ann Arbor, MI.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Doran, C. and Bangalore, S. (1994). Bootstrapping a Wide-coverage CCG from FB-LTAG. In *Third International Workshop on Tree-Adjoining Grammars (TAG+3)*, TALANA, Paris.

Fordyce, C. (2007). Overview of the IWSLT 2007 Evaluation Campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 1–12, Trento, Italy.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., and Thayer, I. (2006). Scalable Inference and Training of Context-Rich Syntactic Models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia.

Goodman, J. (2001). A Bit of Progress in Language Modeling. *Computer Speech and Language*, **15**(1):403–434.

Goodman, J. (2002). Sequential Conditional Generalized Iterative Scaling. In *40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 9–16, Philadelphia, PA.

Hassan, H., Hearne, M., Way, A., and Simaan, K. (2006). Syntactic Phrase-Based Statistical Machine Translation. In *Proceedings of the IEEE 2006 Workshop on Spoken Language Technology*, pages 238–241, Palm Beach, Aruba.

Hassan, H., Ma, Y., and Way, A. (2007a). MaTrEx: the DCU Machine Translation System for IWSLT 2007. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 69–75, Trento, Italy.

Hassan, H., Simaan, K., and Way, A. (2007b). Integrating Supertags into Phrasebased Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 288–295, Prague, Czech Republic.

Hassan, H., Sima'an, K., and Way, A. (2008a). Syntactically Lexicalized Phrase-Based Statistical Translation. *IEEE Transactions on Audio, Speech and Language Processing*, **6**(7):1260–1273.

Hassan, H., Way, A., and Simaan, K. (2008b). A Syntactic Language Model Based on Incremental CCG Parsing. In *Proceedings of the IEEE 2008 Workshop on Spoken Language Technology*, Goa, India.

Hockenmaier, J. and Steedman, M. (2007). CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, **33**(4):355–396.

Ittycheriah, A. and Roukos, S. (2007). Direct translation model 2. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 57–64, Rochester, NY.

Joshi, A. and Schabes, Y. (1991). Tree Adjoining Grammars and Lexicalized Grammars. In Nivat, M. and Podelski, A., editors, *Tree Automata and Languages*, pages 409–431. Elsevier, Amsterdam, The Netherlands.

Kneser, R. and Ney, H. (1995). Improved backing-off for M-gram language modeling. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 181–184, Detroit, MI.

Knight, K. (1999). Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Koehn, P. (2004a). Pharaoh: A Beam Search Decoder for phrase-based Statistical Machine Translation Models. Machine Translation: From Real Users to Research. In *Proceedings of 6th Conference of the Association for Machine Translation in the Americas, AMTA 2004*, pages 115–124, Washington, DC.

Koehn, P. (2004b). Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, Edmonton, AB, Canada.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit X: The Tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowen, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL 2007 Demo and Poster Session*, pages 177–180, Prague, Czech Republic.

Koehn, P., Och, F., and Marcu, D. (2003). Statistical Phrase-Based Translation. In *Proceedings of the Joint Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 127–133, Edmonton, AB, Canada.

Kumar, S. and Byrne, W. (2005). Local phrase reordering models for statistical machine translation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT '05)*, pages 161–168, Vancouver, BC, Canada.

Marcu, D., Wang, W., Echihabi, A., and Knight, K. (2006). SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 44–52, Sydney, Australia.

Marcus, M., Hindle, D., and Fleck, M. (1983). D-theory: talking about talking about trees. In *Proceedings of the 21st Annual Meeting on Association for Computational Linguistics*, pages 129–136, Cambridge, Massachusetts.

Marcus, M., Santorini, B., and Marcinkiewicz., M.-A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, **19**(2):313–330.

Marslen-Wilson, W. (1973). Linguistic structure and speech shadowing at very short latencies. *Nature*, **244**:522–533.

Milward, D. (1994a). Dynamic Dependency Grammar. *Linguistics and Philosophy*, **17**:561–605.

Milward, D. (1994b). Non-Constituent Coordination: Theory and Practice. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, pages 935–941, Kyoto, Japan.

Milward, D. (1995). Incremental Interpretation of Categorial Grammar. In *Proceedings of 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL-95)*, pages 119–126, Dublin, Ireland.

Nasr, A. and Rambow, O. (2004). Supertagging and full parsing. In *Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*, pages 57–63, Vancouver, BC, Canada.

Nivre, J. (2004). Incrementality in Deterministic Dependency Parsing. In *Proceedings of the ACL Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain.

Och, F. (2003). Minimum Error Rate Training in Statistical Machine Translation. In *41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Och, F., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., and Radev, D. (2004). A smorgasbord of features for statistical machine translation. In *Proceedings Human-Language Technology and North American Association of Computational Linguistics (HLT-NAACL),* pages 161–168, Boston, MA.

Och, F. and Ney, H. (2002). Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, PA.

Och, F. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, **29**:19–51.

Owczarzak, K., Genabith, J., and Way, A. (2007). Dependency-based automatic evaluation for machine translation. In *In Proceedings of HLT-NAACL 2007 Workshop on Syntax and Structure in Statistical Translation*, pages 57–64, Rochester, NY.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, **32**(1):71–106.

Papineni, K., Roukos, S., and Ward, T. (1997). Feature-based language understanding. In *Proceedings of 5th European Conference on Speech Communication and Technology EUROSPEECH '97,* pages 1435–1438, Rhodes, Greece.

Papineni, K., Roukos, S., and Ward, T. (1998). Maximum likelihood and discriminative training of direct translation models. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 189–192, Seatle, WA.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 311–318, Philadelphia, PA.

Patry, A., Gotti, F., and Langlais, P. (2006). MOOD: A Modular Object-Oriented Decoder for Statistical Machine Translation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 709–714, Genoa, Italy.

Quirk, C., Menezes, A., and Cherry, C. (2005). Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 271–279, Ann Arbor, MI.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2):257–286.

Ratnaparkhi, A. (1997). A linear observed time statistical parser based on maximum entropy models. In *Proceedings of Second Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–10, Providence, RI.

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

Sagae, K. and Lavie, A. (2006). A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 691–698, Sydney, Australia.

Shen, L. and Joshi, A. (2005). Incremental LTAG parsing. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, pages 811–818, Vancouver, BC, Canada.

Shen, L., Xu, J., and Weischedel, R. (2008). A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, OH.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA.

Stolcke, A. (2002). SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the 7th International Conference on Spoken Language*, pages 901–904, Denver, CO.

Sturt, P. and Lombardo, V. (2004). Processing coordinated structures: Incrementality and connectedness. *Cognitive Science*, **29**(2):291–305.

Tillmann, C. (2004). A unigram orientation model for statistical machine translation. In *Proceedings Human-Language Technology and North American Association of Computational Linguistics (HLT-NAACL)*, pages 101–104, Boston, MA.

Tillmann, C. and Ney, H. (2003). Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133.

Tillmann, C. and Xia, F. (2003). A Phrase-based Unigram Model for Statistical Machine Translation. In *Proceedings of the Joint Meeting of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 106–108, Edmonton, AB, Canada.

Vauquois, B. (1968). A Survey of Formal Grammars and Algorithms for Recognition and Transformation in Machine Translation. In *IFIP Congress-68*, pages 254–260, Edinburgh, UK.

Wang, W., Stolcke, A., and Harper, M. (2004). The use of a linguistically motivated language model in conversational speech recognition. In *Proceedings of Acoustics, Speech, and Signal Processing (ICASSP)*, pages 261–264, Montreal, Canada.

Weaver, W. (1949). Translation. In Locke, W. and Booth, A., editors, *Machine translation of languages: fourteen essays*, pages 15–23. MIT Press, Cambridge, MA, 1955.

Wu, D. (1997). Stochastic Inversion Transduction Grammars and Bilingual parsing of Parallel Corpora. *Computational Linguistics*, **23**(3):377–403.

Wu, D. (2005). MT model space: Statistical vs. compositional vs. example-based machine translation. *Machine Translation*, **19**:213–227.

Xu, P., Chelba, C., and Jelinek, F. (2002). A study on richer syntactic dependencies for structured language modeling. In *40th Meeting of the Association for Computational Linguistics (ACL'02)*, pages 191–198, Philadelphia, PA.

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency Analysis with Support Vector Machines. In *Proceedings of 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206, Nancy, France.

Yamada, K. and Knight, K. (2001). A Syntax-Based Statistical Translation Model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics and 10th Meeting of the European Chapter*, pages 523–530, Toulouse, France.

Zens, R., Och, F., and Ney, H. (2002). Phrase-based statistical machine translation. In Jarke, M., Koehler, J., and Lakemeyer, G., editors, *25th German Conf. on Artificial Intelligence (KI2002)*, pages 18–32. Lecture Notes in Artificial Intelligence (LNAI), Aachen, Germany.

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation, HLT/NAACL*, pages 138–141, New York, NY.