

Improving Statistical Machine Translation with Target-Side Dependency Syntax

John Walter Richardson

September 2016

Abstract

Natural Language Processing (NLP) is the field of Artificial Intelligence research that aims to teach computers to produce and understand human text and speech. Major applications include information retrieval, human-computer dialogue systems and natural language generation. Machine Translation (MT) is the application of NLP that focuses on the automatic translation between languages.

Machine translation is a valuable tool for both personal and business applications. With the incredible diversity of languages accessible on the internet, automatic translation has become a necessity in this era of information. For example, the popular engine Google Translate is used to translate over 100 billion words¹ per day (as of 2016). While other areas of NLP, for example speech recognition, have already reached a level enabling practical application, there remain many obstacles to producing high-quality automatic translation. In particular, translation is difficult for linguistically distant language pairs such as English–Japanese and for resource-poor languages such as Māori, Welsh and Zulu. In this thesis we focus on the challenges of translating between linguistically distant language pairs and a potential solution known as ‘syntax-based MT’.

Syntax-based MT is an MT paradigm based on the principle of generalizing language with grammar. This additional layer of abstraction enables the design of more flexible translation rules. Source language input is first analyzed syntactically, for example in the form of dependency parses, and this grammatical skeleton is transformed to the target language using prelearned syntactic transfer rules. The syntactic frame is fleshed out by combining translated words and phrases to form a complete translation.

¹<https://googleblog.blogspot.jp/2016/04/ten-years-of-google-translate.html>

The majority of previous approaches to syntax-based MT have employed only source-side grammar (known as ‘tree-to-string MT’). This is mainly because syntactic analysis is difficult, prone to error and resulting systems can become overly complicated. While there have been previous studies on exploiting target-side syntax (‘tree-to-tree MT’), results have not been promising. Our aim is to analyze the effectiveness of target-side syntax in the modern world of machine translation. We ask the question of whether the potential improvement in translation quality is able to outweigh the increased complexity of using a structured target-side representation (in particular, dependency parses).

In Chapter 1, we give an overview of machine translation, outlining the major paradigms and methods of evaluation.

Chapter 2 outlines the case study of a state-of-the-art dependency tree-to-tree system, KyotoEBMT, which we have been developing as a core component of our research on syntax-based MT. In this chapter we discuss the design and extraction of dependency tree-to-tree translation rules. Analysis of the system gives empirical evidence of the advantages and disadvantages of syntax-based approaches and provides a starting point for our investigation.

We proceed to analyze two major aspects of translation where target-side syntax can be effective: word order and translation fluency. We discuss our approaches to each of these areas in Chapters 3, 4 and 5 (fluency), and Chapter 6 (word order). In each chapter we describe experiments assessing the effectiveness of our proposed approaches and discuss the potential impact of each method.

While this thesis concentrates on statistical syntax-based approaches, the field has recently seen a surge in interest in translation methods based on neural networks. Chapter 7 presents an overview of future work that could incorporate ideas from this paradigm. We conclude in Chapter 8 by discussing the potential impact and future directions of our work.

Acknowledgments

I would like to express my sincere gratitude to Professor Sadao Kurohashi for his attentive guidance during the six years I have spent in his laboratory in Kyoto. He has been continually supportive of both my academic and personal objectives, and has helped me to develop valuable research and presentation skills. I would also like to thank the other members of my thesis examination committee, Professor Katsumi Tanaka and Professor Tatsuya Kawahara, for their advice and feedback.

Much of the research presented in this thesis would not have been possible without the expertise of Dr Toshiaki Nakazawa and Dr Fabien Cromières, with whom I have been fortunate to work closely on the KyotoEBMT project. I am also indebted to Professor Daisuke Kawahara and Dr Tomohide Shibata for their advice and guidance throughout my time in Kyoto. I have learned much from discussions with countless researchers in the MT community, in particular Professor Graham Neubig, Dr Chenhui Chu and Mr Raj Dabre.

The work in Chapters 4 and 5 was conducted during two internships at Google Japan under the patient and inspiring supervision of Dr Taku Kudo and Dr Hideto Kazawa. I would like to thank the many other Googlers who helped with this project. I am also very grateful to the Japanese Ministry of Education, Culture, Sports, Science and Technology for their financial support, and to Ms Yuko Ashihara for her tireless assistance in administrative matters.

Finally I would like to thank my family and friends, especially my wife Hua Yu for putting up with me during my student years, and Jock and Nancy for their constant encouragement. This thesis is dedicated to my parents, Andrew and Julie, for their love and support over more than 20 years of education.

Contents

Abstract	ii
Acknowledgments	iii
1 Introduction	1
1.1 History of Machine Translation	1
1.2 Major MT Paradigms	2
1.2.1 Rule-Based MT	2
1.2.2 Example-Based Machine Translation	3
1.2.3 Statistical Machine Translation	4
1.2.4 Neural Machine Translation	8
1.3 Overview of Syntax-Based MT	10
1.3.1 Constituency and Dependency Grammars	10
1.3.2 Classification of Syntax-Based Models	12
1.4 Machine Translation Evaluation	12
1.4.1 Automatic Metrics: BLEU and RIBES	13
1.4.2 A Human Metric: Crowd-Sourcing	14
1.5 Thesis Overview	15
2 KyotoEBMT: A Dependency Tree SMT System	17
2.1 Introduction	17
2.2 Historical Background	18
2.3 System Overview	19
2.3.1 Translation Pipeline	19

2.3.2	Example of the Translation Process	21
2.3.3	Implementation Details	21
2.4	Translation Rules	24
2.4.1	Alignment	24
2.4.2	Example Database and Retrieval	26
2.5	Decoding	28
2.5.1	Lattice Decoder	28
2.5.2	Language Model	29
2.5.3	Features and Tuning	30
2.6	Experiments	30
2.6.1	Experimental Settings	30
2.6.2	Results	32
2.6.3	Error Analysis	34
2.7	Open-Source Release	35
2.8	Conclusion	37
3	Post-Editing with Dependency Syntax	39
3.1	Introduction	39
3.2	Related Work	40
3.3	Syntax-Based Post-Editing	42
3.3.1	Operations	42
3.3.2	Filtering Replacements/Deletions with Word Alignments	43
3.4	Experiments	45
3.4.1	Data and Settings	45
3.4.2	Evaluation	45
3.4.3	Tuning and Test Experiments	46
3.5	Error Analysis and Conclusion	47
3.6	Summary	49
4	Dependency Tree Language Model I	50
4.1	Introduction	50
4.2	Related Work	52
4.3	Model Details	53

4.3.1	Classic n -grams with Linear History	53
4.3.2	Syntax-Based History and t -treelets	54
4.3.3	Smoothing	58
4.3.4	Application to SMT: Filtering and Reranking	58
4.4	Experimental Setup	59
4.4.1	Language Choice	59
4.4.2	Automatic Evaluation Metrics	60
4.4.3	Training and Lookup	61
4.4.4	Reranking SMT Output	62
4.5	Optimization of Model Parameters	62
4.5.1	Results	63
4.6	Final Evaluation and Error Analysis	65
4.6.1	Experimental Settings	65
4.6.2	Human Evaluation	65
4.6.3	Results	66
4.6.4	Discussion	67
4.6.5	Error Categorization	68
4.7	Example Sentences (Improved)	70
4.8	Example Sentences (Worsened)	71
4.9	Summary	73
5	Dependency Tree Language Model II	75
5.1	Introduction	76
5.2	Syntactic Transfer	77
5.2.1	Related Work	77
5.2.2	Source-to-Target Dependency Mapping	78
5.3	Projected Language Model	79
5.3.1	Word Context	79
5.3.2	Exploiting Linguistic Knowledge	80
5.3.3	Language Model Training	81
5.3.4	Post-editing	81
5.4	Experiments	82

5.4.1	Filtering	83
5.4.2	Evaluation	83
5.4.3	Results and Error Analysis	84
5.5	Improving Coverage with Delexicalization	86
5.5.1	Change Rate Analysis	88
5.5.2	Delexicalization	88
5.6	Summary	89
6	Dependency T2T Reordering	91
6.1	Introduction	92
6.2	Dependency Tree-To-Tree Translation	93
6.3	Flexible Non-Terminals	95
6.3.1	Rule Augmentation	96
6.3.2	Features	96
6.3.3	Decoding	99
6.4	Experiments	102
6.4.1	Data and Settings	102
6.4.2	Evaluation	102
6.5	Discussion and Error Analysis	103
6.5.1	Non-Terminal Matching Analysis	104
6.5.2	Translation Examples	105
6.6	Summary	106
7	Neural Networks: The Future?	107
7.1	Introduction	107
7.1.1	Language Models	107
7.1.2	Translation Models	109
7.1.3	Hybrid Syntax with Recursive NNs	110
7.1.4	Overview	111
7.2	Neural Networks for Reranking	111
7.2.1	Reranking Framework	112
7.2.2	Features	112
7.2.3	Experiments	113

7.2.4	Conclusion	113
7.3	Improving Flexible Non-Terminals with NNs	115
7.3.1	Insertion Position Selection	115
7.3.2	Experiments	117
7.3.3	Conclusion	117
7.4	Summary	118
8	Conclusion	120
8.1	Overview	120
8.2	Future Work	122
8.2.1	Traditional Approaches	122
8.2.2	Neural Networks	122
8.2.3	Final Words	123
	Bibliography	124
	List of Major Publications	139
	List of Other Publications	140

List of Figures

1.1	EBMT: Comparing examples to infer translation rules. For similar translations ‘ $abc \rightarrow a'b'c'$ ’ and ‘ $adc \rightarrow a'd'c'$ ’ we can infer the rules ‘ $a[X]c \rightarrow a'[X]c'$ ’, ‘ $b \rightarrow b'$ ’ and ‘ $d \rightarrow d'$ ’, where $[X]$ denotes a placeholder non-terminal. This figure is based on Figure 1 from the original paper [64].	3
1.2	Artificial neuron used in neural networks. The neuron calculates a linear combination of inputs x with weights w and filters the output using an activation function f	9
1.3	Example of a constituency parse, or phrase structure grammar derivation. The first row shows words (terminals) with their parts-of-speech. The parts-of-speech form the first layer of non-terminals and are combined in a bottom-up fashion to progressively longer phrase units. The final row ‘S’ is the completed sentence.	11
1.4	Example of a dependency parse. Edges point from words to their heads and are labeled with dependency relation types.	11
2.1	Translation pipeline. An example database (or ‘translation memory’) is first constructed from a parallel corpus. Translation is performed by the decoder, which combines initial hypotheses generated by the example retrieval module. Weights can be improved with batch tuning.	20

- 2.2 An example of the process of translation of the input sentence ‘彼女
は機械翻訳の教科書を読んだ’ into English (‘She read a textbook on
machine translation’). The input and output trees are shown on the
left and right sides respectively with token IDs in square brackets.
The central section shows matched treelets in the example database.
Non-matching terminals from the full examples are crossed out and
replaced with non-terminals matching input position n and marked
[Xn]. Optional target-side words marked with an asterisk. 22
- 2.3 A screenshot of the web interface showing a Japanese-English trans-
lation. The interface displays the input and translation dependency
trees, as well as the list of examples used with their alignments. The
web interface facilitates easy and intuitive error analysis, and can
be used as a tool for computer-aided translation. 23
- 2.4 Example of alignment results. Black squares show system output
and the filled areas show gold alignments. 24
- 2.5 Example of bilingual treelet pairs that can be extraction from an
aligned and parsed parallel sentence. 27
- 2.6 A translation hypothesis encoded as a lattice. This representa-
tion allows us to handle efficiently the various options encoded in
our translation rules. Paths in this lattice correspond to different
choices of insertion position for non-terminal ‘X2’, morphological
forms of ‘be’, and the optional insertion of ‘at’. 29
- 2.7 Increase in translation quality (BLEU) for official releases of Ky-
otoEBMT. The Moses baseline score is included for comparison. . . 36
- 3.1 String vs tree output. The intended meaning of a translation is
often unclear from only string output. In this example, we cannot
tell easily that ‘translate documents’ is a relative clause (missing
the relative pronoun ‘which’ or ‘that’) and that ‘the paper’ is a
prepositional phrase (missing the preposition ‘in’) rather than the
direct object of ‘described’. 41

- 4.1 Example of long-distance agreement error in a French translation. The grammatical gender of the adjective *utilisées* (f.pl.) should be corrected to *utilisés* (m.pl.) to agree with the noun *minéraux* (m.pl.). The verb *sont* also demonstrates long-distance agreement that would be difficult to capture with a classic n -gram model. . . . 52
- 4.2 Example of t -treelet extraction. The left figure shows an example sentence $\{S, T\}$ and the right figure shows the history H for ‘mice’. The five possible t -treelets of order $l \leq 3$ are shown beneath. 55
- 4.3 The shapes of all possible t -treelet types for $l \leq 3$ with natural language examples and word-by-word glosses. The words marked with dark nodes represent the w_i around which the t -treelets are centered. 56
- 5.1 Post-editing an incorrect word form. Firstly the context is extracted for the word in question (‘Hund’) on both source (green) and target (blue) sides using mapping based on the word alignment (shown with double-headed arrows). ‘Hund’ is replaced by ‘Hundes’, the word with the highest probability of having the marked bilingual context according to the projected dependency language model. . . . 78
- 5.2 Definition of the three types of context: ‘Noun-like’, ‘Adjective-like’ and ‘Verb-like’. The green and blue boxes denote respectively the source and target context for the word marked in red. Dashed lines indicate optional context for languages displaying case agreement. . . . 80
- 6.1 Examples of tree-to-tree translation rules extracted from an aligned and parsed bitext. Colored boxes represent aligned phrases and $[X]$ is a non-terminal. 93
- 6.2 Combination of translation rules, demonstrating non-terminal substitution and multiple possible insertion positions for a non-matching input phrase (‘昨日’). 94

6.3	Possible insertion positions for flexible non-terminals with target-side head ‘read’. Allowed positions are shown in green and disallowed positions are shown in red. We do not allow insertion position 3 because it could allow a non-projective dependency structure.	95
6.4	Example of translation rule with flexible non-terminals generated from the first parallel sentence in Figure 6.1. [X] has a fixed position (4) but [Y] can have multiple positions (1, 3, 5). Each position has an associated set of features shown in curly brackets, where $\theta_{i,j}$ is the j th feature for insertion position i . The first feature (0 or 1) shows whether the insertion position is unambiguous.	97
6.5	Example showing how a rule containing many flexible non-terminals is encoded into lattice form for decoding.	101
7.1	Structure of a RNNLM. The sequence of word embeddings w_0, \dots, w_{t-1}, w_t are combined recursively to form a state s_t at time t . The same matrices W and U are used for each combination.	108
7.2	Example of a sequence-to-sequence translation model. Target words t_0, t_1 (red) are generated from internal states (green), which in turn are the result of combining source words s_0, s_1 (blue) with already translated target words.	109
7.3	Building a syntactically motivated sentence state with a recursive neural network. Words are combined hierarchically with the same structure as a precomputed syntax tree. In this example we use the binarized constituency parse: (S (NP (DT the) (NN sun)) (VP (VBP shone) (ADVP (RB brightly)))).	111
7.4	Neural network for insertion position selection. The numbers inside boxes show the dimensions of the corresponding vectors.	116

List of Tables

1.1	Rating guidelines for human evaluation.	15
2.1	Comparison of alignment quality after each step of our alignment pipeline.	25
2.2	Number of sentences for each data fold in ASPEC corpus.	31
2.3	Official BLEU/RIBES evaluation results for KyotoEBMT submissions (WAT14 and WAT15).	33
2.4	Categorization of 30 translation errors for each language pair.	34
2.5	Summary of major open-source releases.	37
3.1	Results of tuning experiments on development set.	47
3.2	Final evaluation results on unseen data.	47
3.3	Examples of improved translations after deleting and replacing incorrect function words.	48
3.4	Examples of worsened translations. The first example shows a case where an important function word is lost, and this example was fixed by using the source–target alignments. The second example shows an error caused by model sparsity.	48
4.1	Word agreement/ordering characteristics of the nine languages selected for translation experiments.	60
4.2	Result of varying model size by changing t -treelet filtering threshold f in training. These results used the setting ‘AllTypes’.	61

4.3	Comparison of model formulations enabling various t -treelet types. BLEU and win-rate are shown for each proposed system. The best results are shown in bold type.	64
4.4	Human evaluation results, comparing mean difference between proposed and baseline systems, sorted by language group.	66
4.5	Number of test sentences with each score difference (-6 to +6) between proposed and baseline systems.	67
4.6	Error categories for analyzed test sentences.	69
5.1	Mean sentence-level improvement compared to baseline with confidence interval for $p < 0.05$. All results are significantly positive. . .	85
5.2	Error categorization for sentences worsened by the proposed post-editing method.	86
5.3	Categorization of baseline errors not improved by the proposed system.	86
5.4	Coverage of unigrams and full contexts using Wiktionary, our lexicalized model and delexicalized model. Coverage is defined as the percentage of LM lookups finding matches at inference time.	87
5.5	Human evaluation results of delexicalized model with original lexicalized model results for comparison. The coverage is greatly improved for similar translation quality.	87
6.1	Automatic evaluation of translation quality (BLEU). The results marked with † are significantly higher than the baseline system and those marked with ‡ are significantly higher than the proposed system with no insertion position features ('Flexible'). Significance was calculated with bootstrapping for $p < 0.05$	102
6.2	Automatic evaluation of translation quality (RIBES). The results marked with † are significantly higher than the baseline system and those marked with ‡ are significantly higher than the proposed system with no insertion position features ('Flexible'). Significance was calculated with bootstrapping for $p < 0.05$	103
6.3	Results of non-terminal (NT) matching analysis.	104

7.1	Evaluation results (BLEU/RIBES) for our WAT14/WAT15 submissions after reranking with neural network features.	114
7.2	Translation results using neural networks to determine non-terminal insertion positions. Bold type signifies results significantly better than the baseline ($p < 0.01$).	118

Chapter 1

Introduction

1.1 History of Machine Translation

Translation remains a difficult and time-consuming task. Despite modern advances in AI, we still require skilled humans to produce high quality translations. While in the past translation was restricted to low volumes and limited domains, especially in commercial, military and scientific applications, the modern information era has seen a huge increase in demand for translation. The world-wide web has enabled unprecedented volumes of data to be exchanged on a global scale, and it has become vital to be able to unlock this wealth of multilingual information.

Machine Translation (MT) is the area of Natural Language Processing (NLP) research that aims to translate text and speech automatically from one language to another. A smorgasbord of approaches have been essayed over the years (with varying success), with applications ranging from Cold War military intelligence to social media.

The first well-documented attempt at automatic translation was the 1954 Georgetown-IBM experiment [55], which aimed to translate Russian sentences automatically into English. Despite using a very basic rule-based approach, researchers believed that the problem of translation would be solved within a few years. This was shown not to be the case, however, when 12 years later the ALPAC report [76] recounted the lack of any major progress in the field. This led to a decrease in funding and activity in automatic translation for the next two

decades.

The 1980s and 1990s witnessed a surge in computational power, which allowed for more sophisticated approaches than the early rule-based systems. Two of the most influential attempts were by IBM with their early word-based statistical approach [8] and Makoto Nagao with example-based translation [64].

MT research in the 21st Century has been dominated by phrase-based statistical MT (PBSMT) [50], a development of the word-based IBM approach, syntax-based approaches such as Hiero [17] and most recently sequence-to-sequence translation with neural-networks [97, 3].

1.2 Major MT Paradigms

In this section we give a more detailed summary of the major MT paradigms and how our research fits into the body of previous work.

1.2.1 Rule-Based MT

The first, and arguably simplest, translation paradigm is that of Rule-Based MT (RBMT). A collection of translation rules (usually hand-written) are applied deterministically to an input sentence to transform the source language structure into the target language. These rules can contain placeholders (or ‘non-terminals’) that are later filled with a combination of other rules and word/phrase translations (or ‘terminals’) taken from a dictionary or lexical database.

The major advantage of this approach is its simplicity and flexibility. Almost any rules can be constructed, however it can be unclear how to design such rules most effectively. In particular, hand-written rules are often unable to deal with the many linguistic exceptions apparent in natural languages, and additional rules must be written for all special cases. This makes RBMT extremely challenging to develop and maintain, particularly if multiple language pairs are to be supported. While RBMT was by far the most popular approach in the early decades of MT, it is now impractical to maintain for web-scale data.

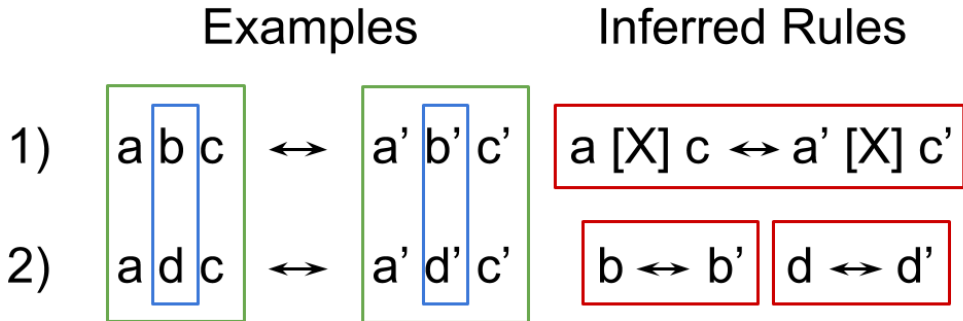


Figure 1.1: EBMT: Comparing examples to infer translation rules. For similar translations ‘ $abc \rightarrow a'b'c'$ ’ and ‘ $adc \rightarrow a'd'c'$ ’ we can infer the rules ‘ $a[X]c \rightarrow a'[X]c'$ ’, ‘ $b \rightarrow b'$ ’ and ‘ $d \rightarrow d'$ ’, where $[X]$ denotes a placeholder non-terminal. This figure is based on Figure 1 from the original paper [64].

1.2.2 Example-Based Machine Translation

Example-Based Machine Translation (EBMT) [64] was designed in an attempt to teach computers to mimic the way humans translate. Nagao observed that humans learn foreign languages by inferring grammar and vocabulary from example sentences, and are able to use semantic understanding to cluster similar words and phrases. EBMT relies on this ‘analogy principle’.

In the original paper, translation rules are extracted from example sentences automatically as shown in Figure 1.1. Substitution rules can be extracted by comparing the source and target sides of two similar translations, and phrase translations can be taken from the differing fragments. These rules can be extended with translations from a dictionary and similar words replaced using a thesaurus.

EBMT received some attention towards the end of the 20th Century, however it was surpassed by statistical approaches, primarily due to its rule-based nature. One of the assumptions was that “language data and its usage do not change for a long time” [64], a premise that no longer applies to the rapidly changing digital world. Owing to its analogical foundations, EBMT is effective for translating sentences very similar to training examples, however can be considerably less effective

for out-of-domain translations.

1.2.3 Statistical Machine Translation

This thesis concentrates primarily on Statistical Machine Translation (SMT). The core concept of SMT is the noisy-channel model [87], which considers translation as the process of reversing the ‘encoding’ of a sentence into another language. This encoding can be seen as the application of random noise sampled from some probability distribution.

In the classical formulation, we write the probability of obtaining a target language translation e from a source language input f as $P(e | f)$, then apply Bayes’ rule to reformulate in terms of reversing the noisy encoding, expressing the probability in terms of $P(f | e)$. We can recover the best translation e^* as follows:

$$\begin{aligned}
 e^* &= \operatorname{argmax}_e P(e | f) \\
 &= \operatorname{argmax}_e \frac{P(f | e)P(e)}{P(f)} \\
 &= \operatorname{argmax}_e P(f | e)P(e)
 \end{aligned}
 \tag{1.1}$$

The formula for e^* contains two parts: $P(f | e)$ and $P(e)$. The former is known as the ‘translation model’ and represents the faithfulness, or ‘adequacy’ of a translation. The latter is known as the ‘language model’ and models the fluency of a translation in the target language.

Translation Model

In SMT, the translation model is usually trained on a set of bilingual sentence pairs, known as a parallel corpus, using unsupervised learning. Such corpora are often obtained from multilingual proceedings of parliamentary meetings, for example Europarl [48], as well as from scientific and commercial sources. To improve translation quality and domain coverage it is important to maximize the volume of parallel data available, and this fact has led to the development of automatic parallel corpora mining from the web [101].

The translation model describes the probability of translating between source-target phrase pairs. The probability distribution is most commonly learned in an unsupervised fashion, specifically with the Expectation-Maximization (EM) algorithm [26]. Sentence pairs are aligned on a word level between source and target languages using algorithms such as the IBM models [9], which are based on the EM algorithm, and these alignments can then be used to extract a word-based translation model, which can be expanded to phrase level by the technique of phrase extraction [50].

Reordering

It is usual for translation models to consider only the translations of individual phrases and to join these together during decoding. This is because it is not practical to learn generic translation models for entire sentences owing to data sparsity. It is therefore necessary to decide how to order these phrases once they have been translated.

For language pairs such as English–French, where the word orders are fairly similar between source and target languages, it can be enough to translate phrases in the same order as the input sentence then to make minor adjustments, such as swapping adjective and noun pairs. This is the basis of the simplest well-known reordering model, linear distortion [50].

Linear distortion models the probability that a pair of phrases a given distance from each other should be swapped. The linear distance is the only parameter and contextual information is ignored, however this simple approach can be effective for similar language pairs. Instead of considering only swapping, reordering (or ‘distortion’) can be split into three types (monotone, swap, discontinuous) for each phrase pair to form a more sophisticated model known as lexical distortion [99]. This approach however is prone to sparsity problems, in particular for distant language pairs.

For more syntactically distant language pairs, we must learn to make linguistic generalizations, such as switching SVO and SOV clause structure and converting post-modifying prepositional and pre-modifying postpositional phrases. Such generalizations often require syntactically motivated long-distance reordering, and a

number of approaches use source-side syntax to pre-order phrases [105, 54, 68].

Language Models

Language models (LMs) are the second major component of SMT systems. They model the fluency of a translation, estimating whether system output ‘sounds natural’ in the target language. Language models are also a fundamental component of many other NLP systems, such as speech recognition and natural language generation frameworks, which also require accurate measurement of natural language fluency.

Since the early days of NLP, the most popular language models have been designed around the concept of n -gram history. The core concept is that the next word in a fluent sentence should be dependent on its preceding words, i.e. context. While ideally all available context of a word would be considered, in practice this can be intractable because of training data sparsity, and extensive context is often unnecessary. It is therefore common to model the next word in a sentence based on the previous $n - 1$ words (for some n), which is known as an n -gram, or sequence of n words.

Based on this intuition, language models have been modeled classically as $(n - 1)$ th order Markov processes, stochastic sequences whose next value depends on the previous $n - 1$ values. The probability of a sentence (sequence of words) w_1, \dots, w_m is often written as:

$$\begin{aligned} P(w_1, \dots, w_m) &= \prod_i^m P(w_i \mid w_1, \dots, w_{i-1}) \\ &\approx \prod_i^m P(w_i \mid w_{i-n+1}, \dots, w_{i-1}). \end{aligned}$$

This linear n -gram context can be used successfully to model sentence fluency for simple sentences, however n -gram sparsity can remain an issue. For this reason, smoothing techniques such as modified Kneser-Ney [15] and Stupid Backoff [7] are often applied.

Standard LMs can fall short when dealing with long-ranged phenomena such as word agreement that require consideration of non-local context. Syntax-based

LMs [14, 63] and neural network LMs [62, 96] have been proposed to consider a more generalized context and are often combined with traditional SMT systems.

Decoding

Once translation and language models have been trained on parallel data, translation of unseen sentences can be attempted. In order to achieve high translation quality, the entire space of translations must be searched to find the translation e^* with the highest probability. This process is known as ‘decoding’.

Decoding to optimize only the translation model score is simple as the scores of individual phrases are independent, however the language model also introduces dependencies between phrases. A variety of techniques have been employed to overcome this, for example storing limited language model histories (or states) at each search node [52, 38].

The number of possible translations grows exponentially with the number of input sentence words, requiring sophisticated search algorithms to find good translations within a short time. The simplest decoding algorithm with a practical decoding speed is beam search, which restricts the number of translations of phrases of each length to a size k , requiring k^2 translations for each combination of two phrases. This can be further reduced using the cube pruning technique [18], which selects only the most likely of the k^2 combinations considered in beam search.

There are a variety of more sophisticated algorithms in use, such as lattice-based search [25]. Other techniques involve limiting the length of phrases and pruning similar translation fragments.

Post-Editing

While decoding is the final necessary step of translation, it is often not sufficient. The translations generated by most automatic systems often contain errors that are able to be corrected relatively easily after translation in a process known as post-editing.

Post-editing in its simplest form corrects the single best translation generated by the decoder, for example regularizing capitalization and punctuation. It is also possible to generate a k -best list of translations and to rerank these using a

post-editing reranker. A common example is to use a powerful language model to rerank k -best translations. The two approaches can also be combined.

The primary advantages of post-editing are its simplicity and speed. It can be computationally expensive and algorithmically complicated to use complex features during decoding, and it is much faster to calculate such features only k times for reranking. It is also often the case that a good translation is not found by the decoder (search error) even though the translation and language models are able to detect better translations.

The benefits of post-editing are particularly clear in complicated systems. In this thesis we make extensive use of post-editing to overcome the computational difficulties of using complex target-side syntax features during decoding. In some cases post-editing also allows our models to be used even with string-based decoders.

1.2.4 Neural Machine Translation

The most recent approach to translation has been Neural MT (NMT), which is based on artificial neural networks (NNs). Neural networks have existed since the early 50s as an attempt to model biological brains.

The basic unit of a neural network is the artificial neuron. An example of such an artificial neuron is shown in Figure 1.2. Single artificial neurons combine multiple weighted inputs to produce a single output, and these neurons can be joined together into a network structure and often grouped into layers. For n inputs x_i with weights w_i and an activation function f , the output y of an artificial neuron is given by:

$$y = f\left(\sum_{i=1}^n w_i x_i\right) \quad (1.2)$$

The perceptron [83], a learning algorithm for binary classification using artificial neurons, was used historically for a multitude of machine learning tasks. However, the perceptron was found to be less viable for large-scale learning than other other more popular classifiers such as Support Vector Machines (SVMs) [102].

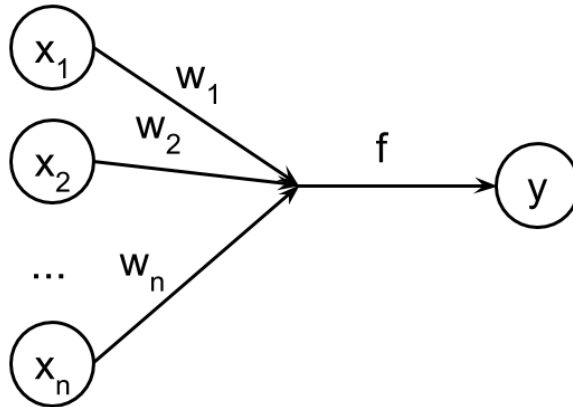


Figure 1.2: Artificial neuron used in neural networks. The neuron calculates a linear combination of inputs x with weights w and filters the output using an activation function f .

Recently the tremendous increase in data and computational resources, combined with parallelized linear algebra computations exploiting Graphics Processing Units (GPUs), has opened the doors to large-scale deep learning. Deep learning is the large-scale combination of neurons into a multilayered (or ‘deep’) network. Now that the training of such large networks has become computationally viable, approaches based on deep learning have produced state-of-the-art results for many AI tasks, from automatic translation [97, 3] to playing the game of Go [92].

Neural networks can be considered similar to a black box that learns to produce some specific output given a set of inputs. The simplest neural translation systems take an input sequence of source sentence tokens and directly output a sequence of translated tokens. These are known as sequence-to-sequence models [97]. The simple sequence-to-sequence model can be improved further by expanding the network to model word alignments and reordering with an ‘attention’ mechanism [3].

A major advantage of NMT is that the long pipeline of independent processes in SMT (alignment, phrase extraction, decoding) can be replaced with a simple

and more elegant model that directly translates input sentences, learning a joint translation and language model. NMT systems have begun to give state-of-the-art results for close language pairs, however at the time of writing are not able to outperform syntax-based approaches for distant language pairs.

1.3 Overview of Syntax-Based MT

This thesis focuses primarily on syntax-based MT, the most linguistically motivated subfield of SMT. In contrast to standard SMT, which takes raw word sequences as input and output, the core concept of syntax-based SMT is to replace raw words with rich tree structures that express the relationships between words. These sentence-level grammatical structures can be used to model more generic rules than standard SMT, for example the difference between SOV and SVO word orders. This makes syntax-based approaches effective at dealing with syntactically distant language pairs.

1.3.1 Constituency and Dependency Grammars

Sentence structure, or syntax, is usually expressed in terms of constituency grammar (also known as ‘phrase structure grammar’) [21] or dependency grammar [98].

Constituency grammar builds sentences from a set of phrase structure rules, combining sentence fragments in a bottom-up fashion using non-terminal symbols that represent phrasal nodes. Dependency grammar instead considers the relationship between words directly by building a graph of labeled directed edges representing word dependencies. Figure 1.3 and Figure 1.4 show an example of a constituency and dependency parse for the same sentence.

There are a number of major differences between these two forms of syntax. Perhaps the most significant difference is that dependency parses do not indicate word order, allowing for more flexibility to express word relations that are position independent. Furthermore, words can have multiple dependents in a dependency parse, however constituency parses are usually binarized, i.e. there are exactly two children for each non-terminal node. This adds flexibility but also complications to dependency syntax.

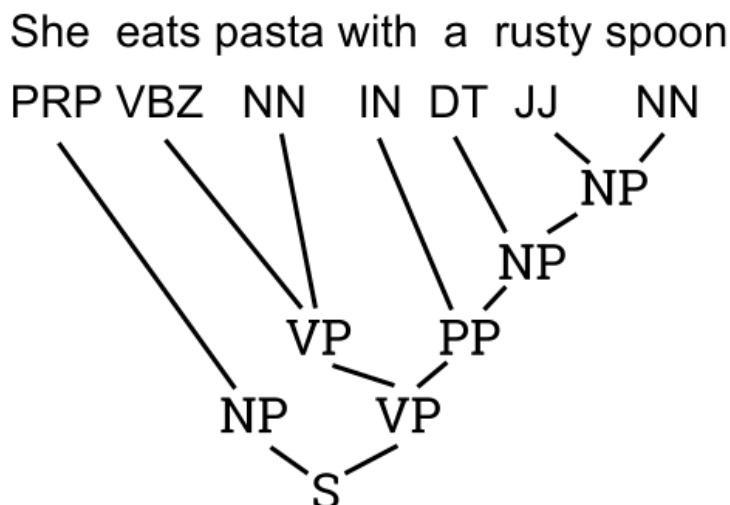


Figure 1.3: Example of a constituency parse, or phrase structure grammar derivation. The first row shows words (terminals) with their parts-of-speech. The parts-of-speech form the first layer of non-terminals and are combined in a bottom-up fashion to progressively longer phrase units. The final row ‘S’ is the completed sentence.

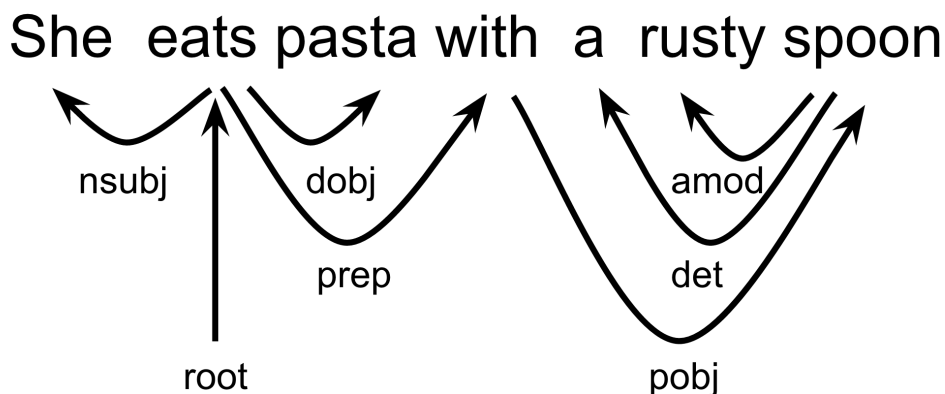


Figure 1.4: Example of a dependency parse. Edges point from words to their heads and are labeled with dependency relation types.

In this thesis we concentrate primarily on dependency parses as we believe they are more expressive and more appropriate for languages such as Japanese and Russian with less rigid word order. We make use of the fact that the relationship between words is clearer than in phrase structure grammar in our work on morphological agreement.

1.3.2 Classification of Syntax-Based Models

Syntax-based MT replaces strings with trees. This can be done for both the source and target sides of a translation. There are therefore four major types of syntax-based models: string-to-string [50, 17], string-to-tree [31, 88], tree-to-string [78, 42, 54] and tree-to-tree [108, 79].

An extension of tree-based systems are forest-based systems [60]. The most serious problems with tree-based approaches are caused by poor parsing quality, which is dependent on a number of factors such as the availability of treebank data and the inherent parsing difficulty of a given language. To help mitigate issues caused by parse errors, multiple parses can be employed. Parse trees, each weighted with their parse score, can be compacted efficiently into a representation known as a ‘forest’. Forests are often used in the source side of translation systems [68, 79].

The choice between using strings or trees is ultimately a trade-off between simplicity and flexibility. String-based approaches are fast and robust, however can only model localized structure. Tree-based approaches are more powerful when high quality parsing is available, but perform more slowly than string-based approaches and are very fragile to parsing errors.

1.4 Machine Translation Evaluation

One of the most important and most difficult challenges in machine translation research is the definition of an effective measure of translation quality. It is difficult to conduct meaningful research when it is not possible to evaluate accurately the quality of translations or to compare two given systems.

Ideally all translations would be evaluated by human professionals who are flu-

ent in both source and target language, however this is not economically feasible. High quality translation evaluation performed by skilled humans requires considerable time and budget. Furthermore, it is often desirable to be able to obtain a rough estimate of quality in a very short time (on the level of microseconds) when tuning systems automatically. It is for these reasons that automatic measures, such as BLEU [74] and RIBES [44] have been developed.

1.4.1 Automatic Metrics: BLEU and RIBES

By far the most popular automatic translation quality metric is BLEU [74]. It has remained in widespread use for its simplicity and relatively good correlation with human judgment.

BLEU is designed to compare n -grams in a reference translation (gold standard translation produced by a human expert) with the system output to be evaluated. It is based on n -gram precisions, i.e. the proportion of n -grams in the system output that appear in the reference translation. The matches are clipped to the number of occurrences of the n -gram in the reference to avoid overcounting.

In addition, a brevity penalty is calculated to penalize short translations. Such a factor is necessary as the n -gram matching considers only precision and not recall. The brevity penalty BP is defined as follows, where r is the length (number of words) of the reference and c is the length of the system output.

$$BP = \begin{cases} 1, & \text{if } c > r \\ e^{1-\frac{r}{c}}, & \text{if } c \leq r. \end{cases} \quad (1.3)$$

Finally, BLEU is defined as the geometric mean of n -gram precisions p_n multiplied by the brevity penalty. Logarithms are used to optimize and reduce floating point errors caused by multiplication.

$$BLEU = BP \cdot \exp \sum_{i=1}^n w_i \log p_i. \quad (1.4)$$

One of the major weaknesses of BLEU is its ability to evaluate word order effectively, although it does implicitly consider localized word order when matching n -grams. The alternative metric RIBES [44] was designed to evaluate word order

explicitly, however it has not seen much global popularity. RIBES calculates the rank correlation of word positions in the system output and reference. It has been shown to have comparable correlation to human judgment as BLEU for similar languages, and is slightly better for distant language pairs.

While both BLEU and RIBES often display weak positive correlation with human judgments, neither can be considered accurate indicators of translation quality. In particular, automatic metrics have been shown to be ineffective in evaluating syntax-based approaches, especially when evaluating improvements affecting long-range agreements [86, 80].

Another major disadvantage of automatic metrics is the requirement for a reference translation. This is often not available. While BLEU was originally designed to be used with multiple references (and performs better with more), it is very rare that more than one reference is available.

Furthermore, automatic metrics usually require translations and references to be segmented into tokens. It is often unclear how best to do this, especially for languages without whitespace such as Chinese and Thai.

1.4.2 A Human Metric: Crowd-Sourcing

Human evaluation of translation quality is often desirable, however professional translators are expensive. This has led to the increase in popularity of translation evaluation with crowd-sourcing [10]. Crowd-sourcing frameworks connect large networks of part-time human workers, and allow for fast and cheap translation evaluation. Workers are paid considerably less than professional translators, however they often lack any specialized skills and can report unreliable ratings.

In our research we employed human evaluation for our language model experiments, as in our opinion it is particularly important to use native speakers to judge translation fluency. We made a compromise between crowd-sourcing and professional evaluation by using semi-skilled raters, who were bilingual speakers of the language pair they evaluated. There were on average around 20 raters per language, split across the test sentences.

Raters were instructed to give a score on a 7-point scale (between 0 and 6 inclusive) for each sentence, given the translation and original source language

Score	Rating guidelines
0	Nearly all the information is lost between the translation and source. Grammar is irrelevant.
2	The sentence preserves some of the meaning of the source sentence but misses significant parts. Grammar may be poor.
4	The sentence retains most of the meaning of the source sentence. It may have some grammar mistakes.
6	The meaning of the translation is completely consistent with the source, and the grammar is correct.

Table 1.1: Rating guidelines for human evaluation.

sentence. Rating guidelines are shown in Table 1.1.

1.5 Thesis Overview

In this thesis we investigate the effectiveness of target-side dependency syntax in improving the fluency of automatic translations.

Dependency tree-to-tree systems are few and far between. While tree-to-string systems have seen much success over the last decade, target-side syntax has seen little popularity. There are a number of reasons for this, primarily the requirement for two parsers (source and target languages) and the multitude of additional complications involved in building T2T systems. These include tree-to-tree rule extraction, reordering and decoding, especially the integration of syntax-based language models.

We analyze each of these aspects in detail. In Chapter 2, we introduce KyotoEBMT, a state-of-the-art dependency forest-to-tree translation system that we have been developing as a core part of our research. We present an overview of how rule extraction and decoding can be performed effectively in a dependency tree-to-tree setting and conduct error analysis of the proposed system. In Chapter 3, we begin our exploration of target-side syntax with the simplest application:

syntax-based post-editing. We expand this work to consider much more generalized post-editing and language modeling in Chapter 4 and Chapter 5, exploring how language models can be improved with dependency information. In Chapter 6, we discuss approaches for reordering exploiting target-side syntax. These methods allow for full decoder integration and show considerable improvement in word ordering for distant language pairs.

Finally, in Chapter 7 we analyze the place of our research in the rapidly evolving world of deep learning. We ask whether target-side syntax can be useful in hybrid and purely neural translation systems. Chapter 8 summarizes our findings and outlines possible directions for future work.

In summary, the goal of this thesis is to answer the simple question: is target-side syntax worth it? There are many arguments why T2T systems could prove more effective than string-based approaches, however this is yet to be explored thoroughly. Can the rich information provided by target-side dependency syntax improve model expressiveness enough to offset the fragility caused by adding another layer of complexity?

Chapter 2

KyotoEBMT: A Dependency Tree SMT System

We begin our exploration of syntax-based machine translation by presenting the KyotoEBMT framework. This chapter outlines the major advantages and disadvantages of dependency tree-to-tree MT and gives experimental evidence of the value of dependency syntax in improving translation quality. We focus on the most fundamental components of SMT systems: alignment, rule extraction and decoding.

2.1 Introduction

String-to-string translation models have been studied extensively and form the basis of traditional SMT systems. However, dependency syntax can provide an additional layer of abstraction, allowing us to consider more generalized translation rules. Such rules can enable translation systems to generate fluent and accurate translations of complex sentences across distant language pairs.

The KyotoEBMT system makes use of both source and target-side dependency syntax. The dependency tree-to-tree translation paradigm has been relatively unexplored as studies on syntax-based MT have tended to focus on constituency trees rather than dependency trees, and on tree-to-string rather than tree-to-tree approaches. Furthermore, we employ separate dependency parsers for each lan-

guage rather than projecting the dependencies from one language to another as in previous work [78]. The dependency structure information is used end-to-end: for improving alignment quality, constraining translation rule extraction and guiding the decoding.

A further unique characteristic of our system is that it does not rely on pre-computation of translation rules, instead extracting translation rules online. This has the merit of enabling the extraction of arbitrarily large rules, allowing for excellent translations of input sentences similar to those in the training data.

We begin by presenting the historical background of ‘KyotoEBMT’, then proceed to describe the system design and our detailed empirical evaluation.

2.2 Historical Background

The translation system at Kyoto University has been in development for over 10 years. Originally the system was inspired by concepts from the EBMT paradigm, however it has now evolved into a fully statistical syntax-based system. Dictionary-based word translations have been replaced by terms extracted automatically from parallel corpora and POS tags and dependency labels are used in place of a thesaurus.

Early versions were slow and unable to translate long sentences, prompting the establishment of a project to build a redesigned and optimized version in early 2013. This chapter describes our work on this new system over the last three years.

While the name ‘KyotoEBMT’ is mainly historical, we maintain a number of ideas that are true to the EBMT paradigm. The most important of these is the use of dependency structure. In his original description of EBMT, Nagao claimed phrase structure grammar to be “not suitable for the analysis of Japanese, because the word order in Japanese is almost free” [64]. Our motivation for using dependency trees is based on this idea.

Another concept we take from EBMT is the use of flexible translation rules, or ‘examples’. These can be of arbitrary size, however we require strict matching of source and target syntax. We therefore use online example retrieval as opposed to a phrase table, which allows for arbitrary example lookup. This is perhaps the

most significant difference between our framework and standard SMT systems.

Overall, the core algorithms of KyotoEBMT can be considered as fundamentally equivalent to those of a regular syntax-based SMT system. For the sake of clarity, in this thesis we consider it as a standard SMT system and use associated terminology.

2.3 System Overview

2.3.1 Translation Pipeline

The KyotoEBMT framework consists of a number of interlinked components that constitute an end-to-end translation pipeline. This section describes the overall architecture of the translation system.

Figure 2.1 shows the basic structure of the KyotoEBMT translation pipeline. The training process begins with parsing of parallel sentences from the training corpus. Both source and target sides are processed with word segmentation and dependency parsing to create a rich structured representation.

The parsed dependency trees are then aligned with the series of steps described in Section 2.4.1. The alignment stage allows us to extract translation rules based on aligned bilingual tree fragments. These are used to build a syntactic phrase database (‘example database’ or ‘translation memory’ in EBMT terminology) containing ‘treelets’ or ‘examples’ that form the initial hypotheses to be combined during decoding.

Translation is performed by first parsing the input sentence. We found that the quality of the source-side parsing had a large impact on translation quality, however parsing errors are unfortunately unavoidable. In our experiments, Chinese parsing was especially challenging and our Chinese parser still produces a significant number of parsing errors. In order to mitigate this problem, we initially tried using a k -best list of input parses. We found this was somewhat successful but inefficient, and therefore extended the k -best list representation of multiple parses to a more compact and efficient forest representation [60]. For the target side we use a 1-best parse.

The input forest is then matched against the example database, searching

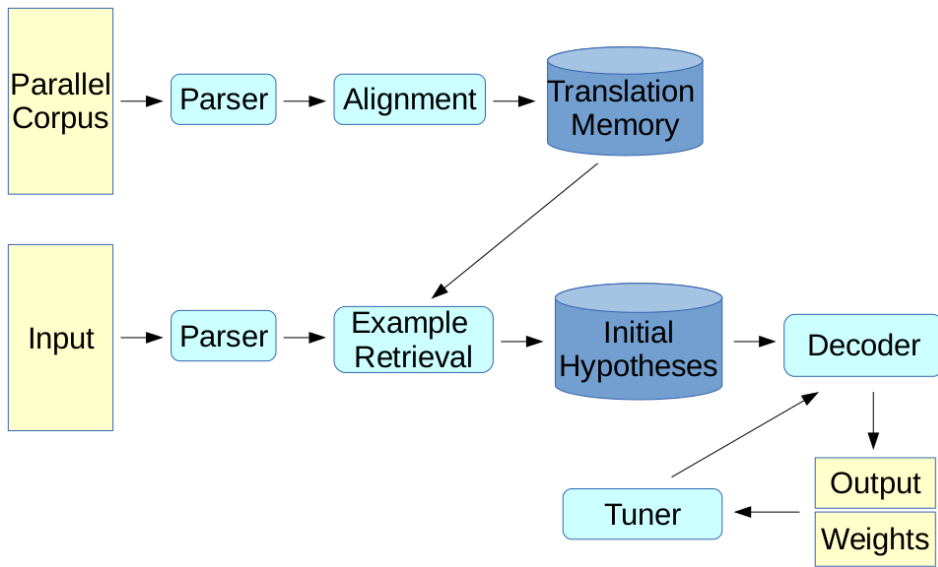


Figure 2.1: Translation pipeline. An example database (or ‘translation memory’) is first constructed from a parallel corpus. Translation is performed by the decoder, which combines initial hypotheses generated by the example retrieval module. Weights can be improved with batch tuning.

for matching treelets. This lookup is performed online using an efficient search algorithm [24]. It is possible to store the retrieved treelets for a given input sentence (‘initial hypotheses’) to disk for optimization purposes. This greatly improves the efficiency of tuning (see Section 2.5.3).

The retrieved treelets are combined by a decoder that optimizes a log linear model score. The decoder uses a lattice structure, integrating partial language model states and considering ‘optional’ words, which are described in Section 2.5. The choice of features and the tuning of the log linear model is described in Section 2.5.3.

2.3.2 Example of the Translation Process

Figure 2.2 shows an example of the translation process. The source sentence (shown on the left) is first parsed. We use dependency forest input, however for clarity only the single best parse appears on the diagram. Matching treelet pairs from the example database are then retrieved. These can be simple word translations such as ‘彼女 → She’ or contain non-terminals, marked with $[Xn]$.

Finally, the translation hypotheses are combined in decoding. This is also when choice of non-terminal positions is made. The right-hand part of the figure shows the derivation of the final translation.

2.3.3 Implementation Details

The system is mostly developed in C++ to optimize translation speed and supports multithreaded decoding. Experiments are facilitated through an end-to-end Experiment Management System (EMS), which is responsible for calculating dependencies between the translation steps, including parsing, alignment, example retrieval, tuning and decoding. Over 90 options for various parameters can be selected on the command line or with a configuration file.

KyotoEBMT also incorporates a web-based translation interface for ease of use. This interface (see Figure 2.3) also displays information required for error analysis such as the input sentence parses and list of translation examples used in decoding.

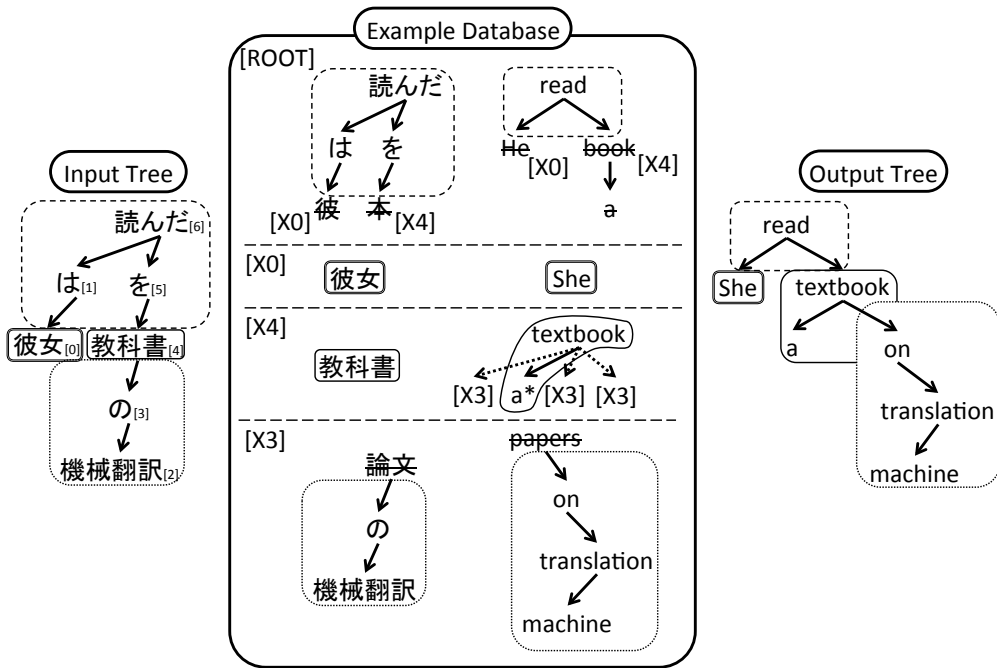




Figure 2.2: An example of the process of translation of the input sentence ‘彼女は機械翻訳の教科書を読んだ’ into English (‘She read a textbook on machine translation’). The input and output trees are shown on the left and right sides respectively with token IDs in square brackets. The central section shows matched treelets in the example database. Non-matching terminals from the full examples are crossed out and replaced with non-terminals matching input position n and marked $[Xn]$. Optional target-side words marked with an asterisk.



本稿では 依存構造 に基づく 用例 ベース 機械翻訳 システム を 紹介する。



Example based machine translation system based on dependency structure are introduced in this paper .

>>

*** Input and Output Dependency Trees ***

0	r[0] 本稿	
1	r[0] で	r[4] an*
2	r[0] は	r[4] example
3	r[6] 依存	r[3] based
4	r[6] 構造	r[2] machine
5	r[5] に	r[2] translation
6	r[5] 基づく	r[1] system
7	r[4] 用例	L[5] based
8	r[3] ベース	H[5] on
9	r[2] 機械	r[6] dependency
10	H[2] 翻訳	L[6] structure
11	r[1] システム	L[5] .*
12	H[1] を	H[0] are*
13	H[0] 紹介	H[0] introduced
14	H[0] する	H[0] in
15	L[7] 。	r[0] this
		L[0] paper
		L[7] .*

*** List of Used Translation Examples ***

[0] NICT JE SP-train-G-0654753		
0	r[0] 本稿	r@[7] in
1	r[0] で	H*[8] ,
2	r[0] は	Hare
3	H@[7] を	H[26] introduced
4	H*[8] ,	Hin
5	H[26] 紹介	r[0] this
6	H[0] した	L[0] paper
7	L*[27] 。	L*[27] .

[1] NICT JE SP-train-R-0064303

0	r@[5] おける	#which
1	r*[6] CAD	L@[8] explains
2	r*[6] /	r@[6] CAD/CAM

Figure 2.3: A screenshot of the web interface showing a Japanese-English translation. The interface displays the input and translation dependency trees, as well as the list of examples used with their alignments. The web interface facilitates easy and intuitive error analysis, and can be used as a tool for computer-aided translation.

└─A						■	■	■	
└─photogate						■	■	■	
is								■	■
└─used								■	■
└─for				■	■				
└─the	■	■	■						
└─photodetector	■	■	■						
	受	光	素子	に	は	フ	ゲ	を	用
						ト	ー		いた

Figure 2.4: Example of alignment results. Black squares show system output and the filled areas show gold alignments.

2.4 Translation Rules

2.4.1 Alignment

The first important stage in generation of translation rules is alignment. We found that alignment accuracy has a major impact on translation quality and therefore use a relatively complex multi-stage alignment process in an attempt to obtain the highest possible accuracy.

Figure 2.4 shows an example of the output of our alignment pipeline. The black squares show final alignments and the filled areas represent the human annotated gold alignments.

In the first stage of alignment, we use the GIZA++ tool [72] to perform word alignment for both source-to-target and target-to-source directions using a combination of IBM models [9] and HMM alignment [103]. The initial alignments are not symmetrized and are of relatively low quality, however it is important to have some initial alignments for the second step.

In the second stage, we perform syntactically motivated alignment using a

	ASPEC-JE			ASPEC-JC		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Stage 1	0.63	0.74	0.68	0.70	0.72	0.71
Stage 2	0.71	0.83	0.76	0.72	0.67	0.70
Stage 3	0.85	0.84	0.85	0.88	0.81	0.84

Table 2.1: Comparison of alignment quality after each step of our alignment pipeline.

Bayesian subtree model based on dependency trees [65]. This contains a tree-based reordering model and can capture non-local reorderings, which sequential word-based models often cannot handle effectively. It also performs monolingual derivations for function words, increasing their alignment accuracy.

In the third stage, we conduct supervised alignment using the Nile [81] tool. Supervised alignment has been shown in previous work [69] to be effective in improving tree-based translation, and we also observed a considerable improvement to translation quality. Since Nile currently supports only constituency parses, we also perform constituency parsing for source and target languages for generating bidirectional word alignments. We use the alignments generated in the second stage as the initial alignments for supervised alignment. Finally, the Nile alignments are post-processed to ensure their compatibility with our dependency-based rule extraction.

Table 2.1 shows a comparison of the alignment quality after each alignment step. The precision, recall and F-score are shown for 100 sentences manually annotated with gold alignment. The sentences were taken from the ASPEC corpus¹ for Japanese–English (ASPEC-JE) and Japanese–Chinese (ASPEC-JC) language pairs. This is the same data we use for the experiments in Section 2.6.

Let the system output alignment links be denoted as A , with gold standard alignments denoted by sure S and possible P links. The precision, recall and F-score are then calculated as follows:

¹<http://orchid.kuee.kyoto-u.ac.jp/ASPEC/>

$$Precision(A, P) = \frac{|P \cap A|}{|A|} \quad (2.1)$$

$$Recall(A, S) = \frac{|S \cap A|}{|S|} \quad (2.2)$$

$$FScore(A, P, S) = \frac{2 \cdot Precision(A, P) \cdot Recall(A, S)}{Precision(A, P) + Recall(A, S)} \quad (2.3)$$

2.4.2 Example Database and Retrieval

An important characteristic of our system is that we do not extract and store translation rules in advance: the process of finding examples partially matching a given input sentence and extracting their translation hypotheses is an online process. A similar approach has been proposed for phrase-based [11], hierarchical [56], and syntax-based [24] systems, however it is rarely seen integrated into syntax-based MT.

This online approach has several benefits. The first is that we are not required to impose a limit on the size of translation hypotheses. Systems extracting rules in advance typically restrict the size and number of extracted rules to avoid becoming unmanageable. A particular advantage is that we will be able to retrieve a near-perfect translation if an input sentence is the same or very similar to one of our translation examples. A second advantage is that we can make use of the full context of the example to assign features and scores to each translation hypothesis.

The main drawback of the online method is that it can be computationally more expensive to retrieve arbitrarily large matchings in the example database online than it is to match pre-computed rules.

We first compact the parsed and aligned training sentences from the parallel corpus into a hypergraph structure. This hypergraph is known as the example database. We use the techniques described in previous work [24] to perform this step as efficiently as possible.

The process of rule construction begins by searching the example database for an example, i.e. an aligned and parsed sentence pair (s, t) , for which s matches a subtree of the input sentence in terms of both word tokens and tree structure. This matched example forms the basis of a translation hypothesis. Figure 2.5

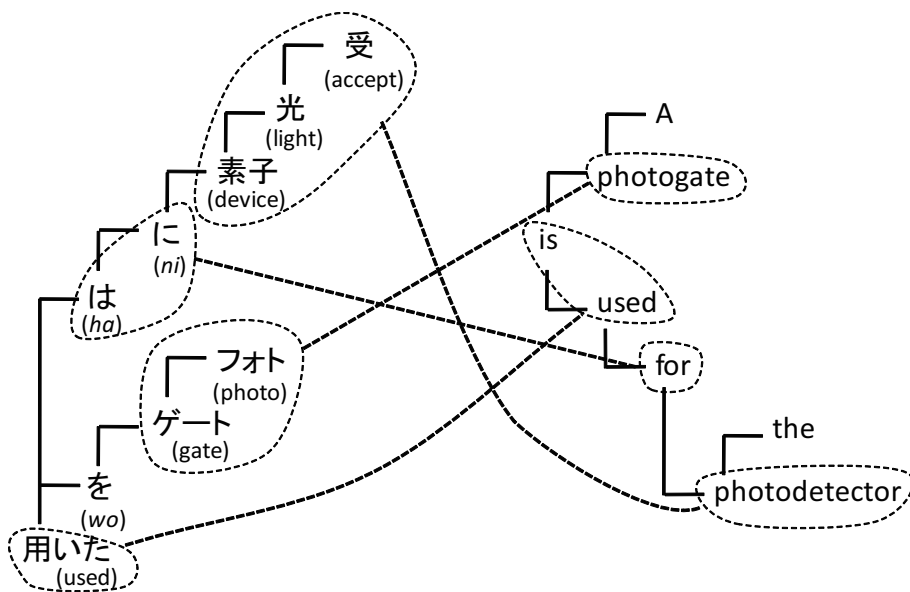


Figure 2.5: Example of bilingual treelet pairs that can be extracted from an aligned and parsed parallel sentence.

illustrates examples of treelet pairs that can be extracted from a sentence in the training corpus.

The target-side of the matched example is extended with non-terminals to represent the insertion positions for the translations of the remaining parts of the input sentence that connect directly to the source-side tree. This is done by replacing unmatched leaves of the example dependency tree with non-terminal symbols. The non-terminal symbols also encode the original input position n (shown as $[Xn]$ in Figure 2.2) to ensure that only the corresponding part of the input sentence is inserted into that non-terminal position.

In some cases it is possible to have multiple positions for non-terminals. This occurs in the case where we have untranslated input tokens but no aligned position on the target side of the rule. We therefore cannot be certain of where the translation should be inserted on the target side, and therefore add multiple candidates for all possible insertion positions. For example, the translation hypothesis containing ‘textbook’ in Figure 2.2 has three possible positions for the non-terminal

matching input position 3. These positions are as a left-side child before ‘a’, a left-side child after ‘a’ or as a right-side child.

This formulation of non-terminals is complicated to handle and does not weight possible insertion points by their likelihood. In Chapter 6, we generalize and expand the model described here to create ‘flexible non-terminals’.

Translation hypotheses can also contain optional target-side words. These are added for null-aligned function words that were not matched in example retrieval, and usually correspond to determiners and punctuation. Optional words are handled by the decoder, which decides whether to include them in the final translation.

Finally we add various features, described in Section 2.5.3, to each translation hypotheses in order to guide the decoding process. Features include statistics about the frequency of occurrence of the matched examples and nature of the non-terminals added.

2.5 Decoding

After having extracted translation hypotheses to cover the entire input tree, we need to decide how to select and combine them to create a complete translation.

2.5.1 Lattice Decoder

The decoder performs a search over the space of rule combinations. This space is constrained by the non-terminal positions, which preserve the structure of the input dependency tree. We use a log linear combination of features to score possible combinations of hypotheses (see Section 2.5.3), and attempt to find the completed sentence that maximizes this model score.

If we only consider local features², then a simple bottom-up dynamic programming approach can efficiently find the optimal combination of a set \mathcal{H} of translation hypotheses with linear $O(|\mathcal{H}|)$ complexity. However, non-local features (such as language model scores) will force us to prune the search space.

²The score of a combination will be the sum of the local scores of each translation hypothesis.

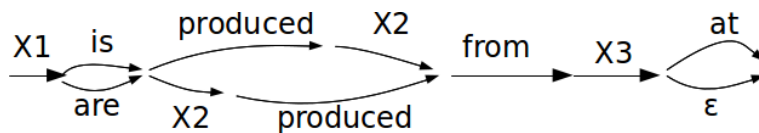


Figure 2.6: A translation hypothesis encoded as a lattice. This representation allows us to handle efficiently the various options encoded in our translation rules. Paths in this lattice correspond to different choices of insertion position for non-terminal ‘X2’, morphological forms of ‘be’, and the optional insertion of ‘at’.

In our experiments we use a lattice-based decoder [25]. As described Section 2.4.2, the translation hypotheses we extract initially from examples have multiple possibilities for non-terminal positions and contain optional words. In order to handle such variations, we use a lattice-based internal representation that can encode them efficiently. An example of such a lattice is shown in Figure 2.6. This lattice representation also allows the decoder to make choices between various morphological variations of a word (e.g. ‘be’/‘is’/‘are’) and such morphological variants can be added in the form of a lookup table.

In addition, our decoder is designed to handle an arbitrary number of non-terminals, which is not considered in previous work such as the original cube-pruning algorithm [18]. This is made simple by the lattice structure, as we simply add sets of edges representing each non-terminal position.

2.5.2 Language Model

For the target-side language model we use KenLM³, a linear 5-gram language model with modified Kneser-Ney smoothing. The target-side tree is compacted into a string representation to calculate the language model score. We use state-reduction [52, 38] and rest-cost estimations [39] to increase the efficiency of handling non-linear language model score combinations.

While allowing for efficient decoding, the use of a standard string-based language model has the disadvantage of not exploiting the target-side syntactic information available to our system. In Chapter 4 and Chapter 5 we explore the

³<http://kheafield.com/code/kenlm/>

effectiveness of using dependency tree language models.

2.5.3 Features and Tuning

We use a log linear model to score each possible combination of hypotheses during decoding. The model consists of a set of features f with associated weights w , and we calculate the model score S as follows:

$$S = \sum_i w_i \log f_i \quad (2.4)$$

We consider a total of 52 features. These can be roughly categorized into three categories: those scoring the quality of example matching, which are local to initial hypotheses; sentence-level features such as language model score; and features describing the path taken by the decoder, for example counting the number of hypotheses combined and optional words selected.

The optimal weights w for each feature are estimated using k -best batch MIRA [16] on a held-out development set. This is a batch tuning algorithm that is able to deal with large numbers of features and works similarly to MIRA, an online margin-based classification algorithm [36, 20], in a batch as opposed to online setting. We found that k -best batch MIRA performed better than other tuning approaches designed for large feature sets, such as PRO [40]. The implementation included in Moses⁴ was used in our experiments.

2.6 Experiments

2.6.1 Experimental Settings

In this section we perform evaluation of the KyotoEBMT system in order to ascertain the strengths and weaknesses of our tree-based approach. Analysis is performed on the experimental results of our submissions to the 1st and 2nd Workshops on Asian Translation [66, 67].

⁴<http://www.statmt.org/moses/>

	JA \leftrightarrow EN	JA \leftrightarrow ZH
Train	2,020,106	667,520
Dev	1,789	2,115
Test	1,812	2,174

Table 2.2: Number of sentences for each data fold in ASPEC corpus.

The experimental data consisted of parallel scientific paper excerpts from the ASPEC⁵ corpus, split into training, development and test folds as shown in Table 2.2. Note that the development and test sets are considerably smaller (relative to the size of the training fold) than those used in other machine learning tasks, such as information retrieval and speech recognition. This is because translation is relatively slow (in the order of 1–10 seconds per sentence) and we require as much of the limited training data as possible. The use of test sets of this size is common practice in MT.

We conducted translation experiments on the four language pairs in the scientific papers subtask: Japanese-English (JA–EN), English-Japanese (EN–JA), Japanese-Chinese (JA–ZH) and Chinese-Japanese (ZH–JA).

The two baseline systems are based on the open-source Moses pipeline with GIZA++ [49]. The baseline system ‘Base-Phrase’ uses the classic phrase-based SMT pipeline [50], while ‘Base-Hiero’ uses the hierarchical phrase-based paradigm [18]. Hierarchical phrase-based SMT attempts to improve upon traditional phrase-based SMT by allowing rules with non-terminal symbols to model long-range phrase reordering. The motivation is similar to syntax-based SMT, however sentence structure is learned in an unsupervised fashion as opposed to relying on independent parsing.

Our systems used the following dependency parsers, which are show below with their approximate parsing accuracies. Accuracies are for unlabeled dependencies and were evaluated by hand on a random subset of sentences from the test data. Note that the parsers were not trained on the same domain (scientific papers) as the translation experiment.

⁵<http://lotus.kuee.kyoto-u.ac.jp/ASPEC/>

- English: NLPParser⁶ (92%) [13]
- Japanese: KNP (96%) [46]
- Chinese: SKP (88%) [89]

For generating input for Nile we used the following constituency parsers:

- English: Berkeley Parser [75]
- Japanese: Cyklark [73]
- Chinese: Berkeley Parser [75]

Our 2014 submission (‘WAT14’) was the basic KyotoEBMT framework using 1-best tree input and the first two alignment stages (GIZA++ and Bayesian subtree alignment).

The 2015 submission (‘WAT15’) improves upon the 2014 submission primarily by using forest input and supervised alignment (all three stages described in Section 2.4.1). Forests were created by packing the 200-best dependency parses for Japanese and English, and 50-best parses for Chinese. The same dependency parsers were used. The feature set was also expanded, in particular adding features considering input parse scores.

2.6.2 Results

Table 2.3 gives the official evaluation results for BLEU and RIBES metrics (see Section 1.4). The results shown are for evaluation on the test set after tuning. Tuning was conducted over 50 iterations on the development set using a 500-best list.

The results show that the WAT15 system achieves the highest translation quality for all language pairs and metrics, with the exception of RIBES for JA–ZH. We can also see a considerable improvement in translation quality between the WAT14 and WAT15 systems.

While it is clear from the automatic evaluation that the proposed system outperforms the baseline systems, it is necessary to analyze the translations in more

⁶Converted to dependency parses with in-house tool.

Language Pair	System	BLEU	RIBES
JA-EN	Base-Phrase	18.45	64.51
	Base-Hiero	18.72	65.11
	WAT14	20.60	70.12
	WAT15	21.31	70.65
EN-JA	Base-Phrase	27.48	68.37
	Base-Hiero	30.19	73.47
	WAT14	29.76	75.21
	WAT15	30.69	76.78
JA-ZH	Base-Phrase	27.96	78.90
	Base-Hiero	27.71	80.91
	WAT14	27.21	79.13
	WAT15	29.99	80.71
ZH-JA	Base-Phrase	34.65	77.25
	Base-Hiero	35.43	81.04
	WAT14	33.57	80.10
	WAT15	36.30	81.97

Table 2.3: Official BLEU/RIBES evaluation results for KyotoEBMT submissions (WAT14 and WAT15).

Error Category	JA-EN	EN-JA	JA-ZH	ZH-JA	Total
Word order	10	5	13	9	37
Fluency	15	7	7	5	34
Alignment	2	11	9	5	27
Parsing	2	3	1	6	12
Word segmentation	1	0	0	4	5
OOV	0	4	0	1	5

Table 2.4: Categorization of 30 translation errors for each language pair.

detail to discover the areas where translation quality requires further improvement. This is discussed in the following section.

2.6.3 Error Analysis

We performed a categorization of 30 errors observed in the output of the WAT15 system for each language pair. The results are shown in Table 2.4.

Overall the most common errors were incorrect word order and word choice (fluency). This is perhaps not surprising for distant language pairs with widely different syntax and vocabulary. The majority of fluency errors were for English as the target language, as there were many morphological (e.g. word agreement) errors.

While Japanese and Chinese display almost no word agreement, mistranslation of Japanese particles was a major cause of loss in translation fluency. The translation below exhibits poor fluency, where the direct object particle ‘を’ is missing at (1) and there is a repeated subject particle ‘が’ at (2).

- **Input:** By converting the phase detected by such a method to angle of projected sheet beam , there can be obtained three-dimensional shapes of the object in real time .
- **Output:** この方法で検出される位相 (1) 角の投影シートビームに変換することにより , 実時間で物体の3次元形状が (2) 得られる。
。

The next most serious issue was word alignment. While we observed quality gains with our three-step alignment approach, the end-to-end translation results show that there is still much room for improvement. The most common issue with alignment was for Japanese particles, which were often mistranslated or omitted. This is because there is often no word-level equivalent in English or Chinese. The example below shows a poor translation caused by misaligned particles. Aligned phrases are marked with (1) and (2):

- **Input:** The motion of water molecules of local parts [which have made (2)] hydrogen bond move collectively [while (1)] water molecules of the other parts do not move .
- **Output:** 水分子の水素結合の動き [を (2)] 一括して局所の水分子 [が (1)] 他の部分の運動には移動しない。

It is common that syntax-based approaches fail due to incorrect upstream monolingual analysis (in particular parsing). These errors were kept to a minimum by employing forest input, however Chinese parsing remains an issue for future improvement. All English parsing errors were caused by incorrect PP attachment. OOV (out-of-vocabulary) errors were most prevalent for English–Japanese translation. This is because the English side of the ASPEC corpus is particularly noisy, containing numerous misspellings and typos, and was not written by native speakers.

The results of our error analysis showed that word order and choice are the areas requiring the most attention. We address function word selection in Chapter 3, then more general word choice in Chapter 4 and Chapter 5, where we attempt to improve morphological agreement in a generalized setting. To improve word order, we propose a novel reordering approach in Chapter 6.

2.7 Open-Source Release

The system has been made available an open-source toolkit in order to promote research on dependency tree-to-tree translation. The code and documentation are

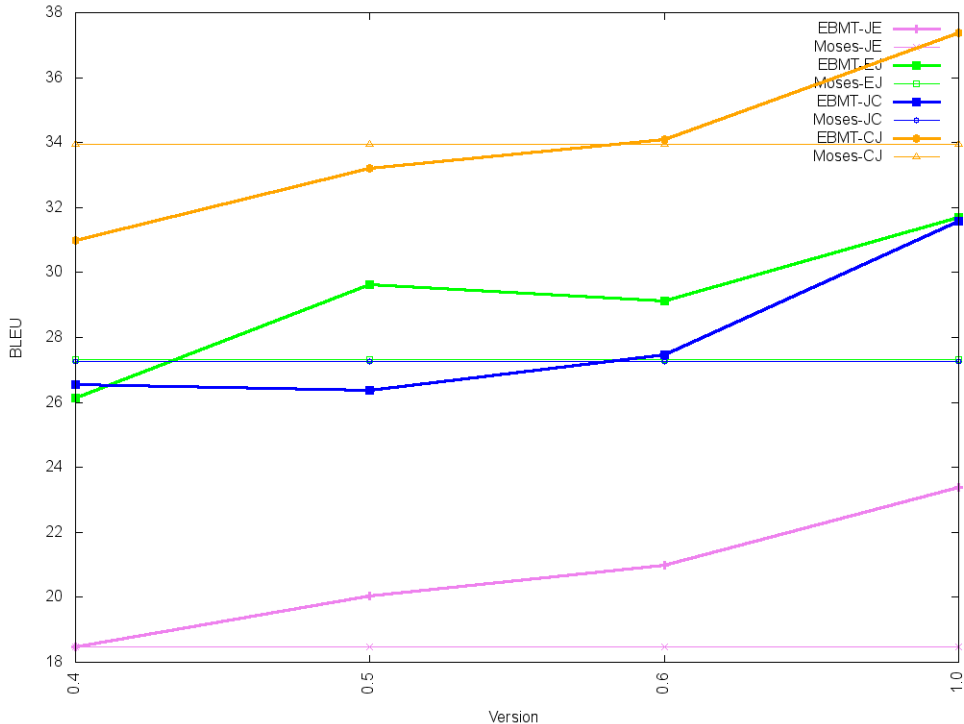


Figure 2.7: Increase in translation quality (BLEU) for official releases of KyotoEBMT. The Moses baseline score is included for comparison.

available⁷ under the AGPLv3 license⁸.

The major changes for each numbered release are summarized in Table 2.5. We have performed on-going evaluation on the ASPEC data (see Section 2.6) for each public release and have seen a constant improvement in translation quality. Figure 2.7 shows how the quality has improved over the last four official releases, along with Moses baseline scores for comparison. Note that these results use slightly different settings from the WAT evaluation in Section 2.6.

⁷<http://lotus.kuee.kyoto-u.ac.jp/~john/kyotoebmt.html>

⁸<https://opensource.org/licenses/AGPL-3.0>

Version	Date	Major Changes
0.1	May 2013	First C++ version
0.2	Aug 2013	Improved decoding, sibling dependency support
0.3	Feb 2014	Lattice decoding, improved features
0.4	June 2014	First public release, tuning improvements
0.5	Oct 2014	Reranking (see Section 7.2), n -best input
0.6	Mar 2015	Expanded feature set, decoder improvements
1.0	Sep 2015	Forest input, improved reranking

Table 2.5: Summary of major open-source releases.

2.8 Conclusion

In this chapter we have described the KyotoEBMT translation system. KyotoEBMT exploits both source and target dependency analysis to improve the quality of translation between distant language pairs. We also employ online example retrieving to enable extraction of arbitrarily large translation examples at translation time.

We believe that the use of dependency syntax is important for accurate translation across distant language pairs, especially in settings such as the WAT translation tasks with many long sentences with convoluted structure. We have designed a complete translation framework around this concept, using dependency trees at each step from alignment to example retrieval to example combination.

Our evaluation demonstrated that the current performance (in terms of BLEU and RIBES) of our system has overtaken state-of-the-art open-source PBSMT systems. Comparison of 2014 and 2015 submissions to the Workshop of Asian Translation showed that improved alignment and the addition of forest input were effective in improving translation quality. Error analysis showed however that parsing and alignment errors do remain. A potential improvement would be to consider using forests for all the translation examples and not simply the input sentence.

Word order and target-side fluency remain serious challenges, as shown by our

error analysis. We address these in the following chapters of this thesis: fluency in Chapters 3, 4 and 5; and word order in Chapter 6.

Chapter 3

Post-Editing with Dependency Syntax

We commence our exploration of target-side syntax with the simplest application: syntax-based post-editing. We show that translation fluency can be improved by exploiting the target-side structure of tree-to-tree machine translation output to post-edit function words automatically.

3.1 Introduction

Post-editing is the process of improving the output of a translation system after decoding has completed. The primary advantages of this approach are its speed and simplicity, as post-editing methods are decoder independent and restrict the search space from the entire hypothesis space to the k -best translation output. In this study we consider 1-best post-editing, i.e. the direct editing of 1-best decoder output, as opposed to the other popular approach of reranking, which attempts to rerank the top k translation candidates.

In our error analysis in Section 2.6.3, we found that word choice and word order are key areas requiring improvement. Indeed, these errors constituted roughly 30% of all mistranslations. In this chapter we attempt to reduce such errors with our proposed syntax-based post-editing approach.

Word choice and ordering errors can cause a significant drop in translation

comprehensibility, especially if function words are omitted or incorrectly translated. In particular, it is important to translate negation and passive structures using correctly placed function words, and prepositional phrases must be correctly ordered and use the appropriate prepositions. We found in our error analysis of current systems that the lack or incorrect placement of relative pronouns has a largely negative effect on preserving sentence meaning, and also that badly formed punctuation impedes understanding.

In the previous chapter we introduced KyotoEBMT, a state-of-the-art dependency tree-to-tree machine translation framework. One of the major advantages of such a system is that we have access to structured output, in the form of a dependency tree, as opposed to a flat string. We believe that this additional structure can help to recover the intended meaning of a poorly translated sentence, and that this is often unclear from flat MT output.

For example, the intended meaning of the translation in Figure 3.1 is much clearer from the dependency tree representation. We can recover the information that ‘translate documents’ is a (mistranslated) relative clause only by inspecting the tree structure of the output. Based on this observation, we consider how it can be possible to use this richer output representation to understand better the cause of function word errors and correct them more effectively.

We propose a post-editing algorithm for editing function words based on a simple dependency tree language model that is able to predict additions, replacements and deletions. We show that a significant improvement in human evaluation can be achieved with our proposed method.

3.2 Related Work

The generation of precise and comprehensible automatic translations remains a considerable challenge. In particular, function words are often poorly translated by standard machine translation systems, particularly across language pairs with greatly differing syntax.

Surprisingly few studies have aimed specifically at improving function word translation for statistical machine translation systems, despite this having been

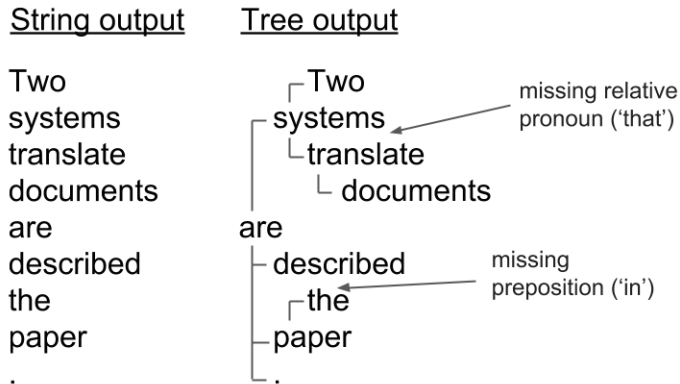


Figure 3.1: String vs tree output. The intended meaning of a translation is often unclear from only string output. In this example, we cannot tell easily that ‘translate documents’ is a relative clause (missing the relative pronoun ‘which’ or ‘that’) and that ‘the paper’ is a prepositional phrase (missing the preposition ‘in’) rather than the direct object of ‘described’.

considered for rule-based systems [1]. While we were unable to find any previous studies specifically on statistical function word post-editing, function words have been exploited to generate improved translation rules in previous work [104].

The most similar approach to our method of editing function words uses structural templates and was proposed for SMT [59]. Statistical post-editing of MT output in a more general sense [93] and learning post-editing rules based on common errors [28, 41] have shown promising results. The majority of statistical post-editing methods work directly with string output, however a syntactically motivated approach has been tried for post-editing verb-noun valency [82].

While a high level of machine translation accuracy is sought after in all subject domains, the correct translation of function words is especially important for patent and scientific translation, where it is necessary for the translation to preserve strictly the meaning of the input sentence. This has led to the promotion of research in this field, especially with shared tasks and workshops on scientific

translation, such as the Workshop on Patent and Scientific Literature Translation¹ and Workshop on Asian Translation [66, 67].

3.3 Syntax-Based Post-Editing

Our proposed method starts with the dependency tree output of a tree-to-tree machine translation system. From this we analyze the position of function words and attempt to modify them with a tree-based function word language model.

We assume a set of function words F , a subset of the entire target-side vocabulary. We also define an empty token ϵ which represents the lack of a function word. A root node and leaf nodes can be added to the tree to allow insertion of function words as the sentence root and leaves respectively.

A dependency tree can be decomposed into token-head pairs (t, t') . We derive a simple language model $P(f | t, t')$ approximating the probability of function word $f \in F$ being inserted between t and t' . The model is estimated over the training data by counting the occurrence of (f, t, t') tuples where f is a function word appearing between t and t' . Note that to make this definition well-defined, we strictly require that function words have only one child. The probability $P(f | t, t')$ is then calculated as:

$$P(f | t, t') = \frac{\text{count}(f, t, t')}{\sum_{g \in F \cup \{\epsilon\}} \text{count}(g, t, t')} \quad (3.1)$$

In our experiments we include part-of-speech tags inside tokens to reduce homonym ambiguity (e.g. use ‘set-NN’ instead of ‘set’). We also split $P(f | t, t')$ into two cases, $P_{\text{left}}(f | t, t')$ and $P_{\text{right}}(f | t, t')$, to consider the difference between t being a left or right descendant of t' . We will write P_s to refer to whichever of P_{left} or P_{right} applies in each case.

3.3.1 Operations

For a token-head pair (t, t') , word insertion is performed when $P_s(f | t, t') > P_s(\epsilon | t, t')$ for some function word f . We choose the function word with the

¹<http://aamtjapio.com/pslt2015/>

highest probability if there are multiple candidates. Replacement of function word t is performed similarly if $P_s(\text{child}(t) \mid f, t') > P_s(\text{child}(t) \mid t, t')$ for some other function word f . Similarly we choose the best f if there are multiple candidates. Deletion can be performed using the same method as for replacement by adding the function word ϵ to F . The full algorithm for post-editing a dependency tree T is shown in Algorithm 1.

3.3.2 Filtering Replacements/Deletions with Word Alignments

In the majority of cases we found it counter-productive to replace or delete function words corresponding directly to non-trivial source words in the input sentence. For example, in a Chinese–English translation task, consider the two translations:

- 听/声音 (listen/sound) \rightarrow listen *to* a sound
- 下面/100/米 (below/100/m) \rightarrow 100m below

In the first sentence, the function words ‘to’ and ‘a’ in the English translation have no corresponding words in the Chinese input and therefore its existence is based only on the target language model. In contrast, the preposition ‘below’ in the second sentence directly corresponds to ‘下面 (below)’ in the input and care should be taken not to delete it (or change it to a completely different preposition such as ‘above’).

We therefore propose restricting replacement/deletion to function words that are aligned to trivial or ambiguous source-side words (function words without concrete meaning, whitespace, punctuation). This allows us to change for instance the unaligned ‘to’ in ‘listen to’ but not ‘below’ with an input alignment. The source–target word alignments are stored in the translation examples used by the baseline SMT system and kept track of during decoding.

Algorithm 1 Post-Edit Tree

```

1: loop:
2: # Traverse tree from left-to-right
3: for  $(t, t') \in T$  do
4:   if  $t \in F$  then
5:      $child \leftarrow \text{GetUniqueChild}(t)$ 
6:     # Find the best function word to replace  $t$ 
7:      $max\_f, max\_p \leftarrow t, P_s(t \mid child, t')$ 
8:     for  $f \in F \cup \{\epsilon\}$  do
9:       if  $P_s(f \mid child, t') > max\_p$  then
10:         $max\_f, max\_p \leftarrow f, P_s(f \mid child, t')$ 
11:       end if
12:     end for
13:     if  $max\_f \neq t$  then
14:       # Replace  $t$  with  $max\_f$  and restart for entire tree
15:        $\text{Tree.Replace}(max\_f, child, t')$ 
16:       goto loop
17:     end if
18:   else
19:      $max\_f, max\_p \leftarrow t, P_s(\epsilon \mid t, t')$ 
20:     # Find the best function word to insert
21:     for  $f \in F$  do
22:       if  $P_s(f \mid t, t') > max\_p$  then
23:         $max\_f, max\_p \leftarrow f, P_s(f \mid t, t')$ 
24:       end if
25:     end for
26:     if  $max\_f \neq \epsilon$  then
27:       # Add function word  $max\_f$  and restart for entire tree
28:        $\text{Tree.Add}(max\_f, t, t')$ 
29:       goto loop
30:     end if
31:   end if
32: end for

```

3.4 Experiments

3.4.1 Data and Settings

We performed translation experiments on the Asian Scientific Paper Excerpt Corpus (ASPEC)² for Japanese–English translation. The data was split into training, development and test sentences as described in Section 2.6.

We defined English function words as those tokens with POS tags of functional types such as determinants and prepositions, and treated Japanese particles as function words for the purposes of alignment-based filtering. The primary post-editing model was trained on the training fold of the ASPEC data. Since our model only requires monolingual data, for comparison we also trained a separate model on a larger (30M sentences) in-house monolingual corpus (Mono) of technical/scientific documents.

KyotoEBMT (see Chapter 2) was used as the baseline SMT system, and post-editing was performed on the top-1 translation. After editing we recalculated the decoder model score (by updating the language model score and word penalty) and used the edited sentence if the model score improved. We found this was slightly more effective than relying solely on the post-edited output.

Japanese segmentation and parsing were performed with JUMAN and KNP [47]. For English we used NLParse [13], converted to dependency parses with an in-house tool. Alignment was performed with the three steps described in Section 2.4.1, including supervised alignment with Nile [81] and our in-house alignment tool based on Gibbs sampling [65]. We used a 5-gram language model with modified Kneser-Ney smoothing built with KenLM [37].

3.4.2 Evaluation

Human evaluation was conducted to evaluate directly the change in translation quality of function words. We found that automatic evaluation metrics such as BLEU [74] were not sufficiently sensitive to changes (the change rate is relatively low for post-editing tasks) and did not accurately measure the function word accuracy.

²<http://lotus.kuee.kyoto-u.ac.jp/ASPEC/>

In human evaluation we asked two native speakers of the target language (English) with knowledge of the source language (Japanese) to decide if the system output was better, worse, or neutral compared to the baseline. A random sample of 5 sets of 20 (development) and 2 sets of 40 (test) edited sentences were selected for each experiment and the identity of the systems was hidden from the raters. The Fleiss’ kappa inter-annotator agreement [30] for wins/losses was 0.663, and when including neutral results this was reduced to 0.285.

3.4.3 Tuning and Test Experiments

We first performed a preliminary tuning experiment on the development fold of ASPEC to investigate the effect of model parameters. The results in Table 3.1 show for each row the model settings, the number of wins (+), losses (−) and neutral (?) results compared to the baseline, and the change rate (CR) over the entire development set.

The first three settings (‘OnlyIns’, ‘OnlyRep’, ‘OnlyDel’) show the effects of allowing only insertions, replacements and deletions respectively without using source–target alignments (see Section 3.3.2). We can see that the quality for deletions is lower than insertions and replacements, and error analysis showed that the major cause was deletion of function words aligned to content words in the input.

We reran the experiments using the alignment-based filtering (‘AlignA’ and ‘AlignB’) and found the results improved. While possible to achieve a higher change rate by allowing all three operations, we could only achieve a slight increase in accuracy by disallowing replacements (the setting ‘AlignB’). The difference was mainly due to alignment errors, which caused more serious problems for replacement as they were able to alter sentence meaning more severely.

The best settings in the tuning experiment (‘AlignB’) were used to conduct the final evaluation on the unseen test data from ASPEC. We also compared models trained on the ASPEC training fold and on our larger monolingual corpus. Table 3.2 shows the final evaluation results. The results on the test set show significant improvement on win/loss sentences at $p < 0.01$. There was no clear improvement gained by increasing the size of model training corpus, however the

change rate could be improved by using more data.

	Insert	Replace	Delete	Align	+	-	?	CR
OnlyIns	Yes	No	No	No	10	6	4	2.3
OnlyRep	No	Yes	No	No	11	7	2	5.5
OnlyDel	No	No	Yes	No	7	8	5	8.6
AlignA	Yes	Yes	Yes	Yes	11	7	2	10.5
AlignB	Yes	No	Yes	Yes	11	4	2	3.3

Table 3.1: Results of tuning experiments on development set.

	Insert	Replace	Delete	Align	+	-	?	CR
ASPEC	Yes	No	Yes	Yes	19	8	13	2.3
Mono	Yes	No	Yes	Yes	23	13	4	4.1
Both	Yes	No	Yes	Yes	42	21	17	3.9

Table 3.2: Final evaluation results on unseen data.

3.5 Error Analysis and Conclusion

The experimental results show that in general our proposed method is effective at improving the comprehensibility of translations by correctly editing function words. Table 3.3 gives examples of improved translations and Table 3.4 shows examples of worsened translations.

We found that using source–target alignments was effective in avoiding errors such as the first example in Table 3.4, however there remained some trickier cases where the alignment information was not sufficient, for example when function words were null or incorrectly aligned. The remainder errors were primarily caused by incorrect parsing and sparsity issues. The second example in Table 3.4 shows such a sparsity error, which could perhaps be fixed by normalizing numerical values.

Input	転倒予防が重視されるのは、大腿骨頸部骨折との因果関係にある。
Baseline	<i>Of</i> fall prevention is emphasized is the causal relation with femoral neck fracture.
Proposed	Fall prevention is emphasized is the causal relation with femoral neck fracture.
Input	今後は、難治性疾患 (...) へと期待が寄せられる。
Baseline	In the future , the expectation is being placed <i>to</i> the treatment of the intractable disease (...).
Proposed	In the future , the expectation is being placed <i>on</i> the treatment of the intractable disease (...).

Table 3.3: Examples of improved translations after deleting and replacing incorrect function words.

Input	特に、簡易比色計によるりん酸塩の測定（モリブデン法）では、(...) 。
Baseline	Especially, <i>in</i> the measurement of phosphate by simple colorimeter (molybdenum method), (...).
Proposed	Especially, the measurement of phosphate by simple colorimeter (molybdenum method), (...).
Input	(...) 小型個体 (...) の水揚げ量を (...) 15%以下に抑えることが 勧告された。
Baseline	(...) it was recommended that (...) suppress fish catch of small individuals (...) <i>to</i> 0,15%.
Proposed	(...) it was recommended that (...) suppress fish catch of small individuals (...) 0,15%.

Table 3.4: Examples of worsened translations. The first example shows a case where an important function word is lost, and this example was fixed by using the source–target alignments. The second example shows an error caused by model sparsity.

3.6 Summary

In our first application of target-side dependency syntax, we have shown the quality of machine translation can be improved through the automatic post-editing of function words using a simple syntax-based function word language model.

We have presented an algorithm for inserting, deleting and replacing function words, and have demonstrated the effectiveness of using source–target alignments to improve accuracy. The results show however that steps must be taken to provide more robustness against parsing/alignment errors. One possible approach would be to consider target-side forests, however it is likely that the increased decoding complexity would also cause search errors.

It is clear from this first simple application that there is promise in using target-side dependency syntax for post-editing, however the scope of this approach requires expanding. In the next chapter we develop the ideas presented here to build a considerably more sophisticated model for generalized post-editing with dependency language models.

Chapter 4

Dependency Tree Language Model I

Our error analysis in Section 2.6.3 showed that poor target-side fluency was the second most prevalent error category for our dependency tree-to-tree system. In the previous chapter we considered a simple approach to solve this using target-side syntax: syntax-based function word post-editing. We develop these ideas in this chapter into a fully generalized dependency tree language model. In order to evaluate the effectiveness of our approach in a more general and challenging setting, we make an in-depth study into improving word agreement for a variety of language families displaying rich morphology.

4.1 Introduction

In this chapter we consider the task of language modeling. Classic n -gram language models are effective at capturing translation fluency at the word level, however such approaches often fail at the syntactic and semantic level. We abstract the traditional definition of a classic n -gram to dependency trees and show how our approach is able to improve more challenging issues such as long-distance word agreement.

The primary motivation for using structured language models is that we can reduce the ‘distance’ between words that are interdependent on a syntactic (and

often semantic) level. While tree-based models have more complicated structure than their string-based counterparts, the sparsity of the most important information is reduced, giving a more compact and relevant representation of context.

Despite recent advances in neural and structured language modeling technology, the most widespread language modeling paradigm in major translation systems is still classic n -gram modeling. Classic n -gram models are combined with a variety of smoothing methods, the most popular being modified Kneser-Ney [15] and Stupid Backoff [7], to form simple and robust models of linear word context. There exist highly optimized implementations, such as KenLM [37], making n -gram models popular in modern systems [49, 68, 79].

There have been a number of issues that have prevented the widespread adoption of tree-based language models. We believe that the two main problems are the lack of an agreed standard on the most effective definition of tree-based context, and the requirement for a syntax-based decoder for full integration into a machine translation system. The use of parsers as language models has shown little improvement in translation experiments [71, 77] and there have been previous attempts to use syntax-based language modeling that have failed to show any statistically significant increase in BLEU [84] (see errata¹). We show that it is still possible to achieve a significant improvement in translation quality judged by humans without requiring a syntax-based decoder.

In the following two chapters we frequently refer to the problem of long-distance word agreement. We use the term ‘long-distance’ to refer to agreements that would not usually be captured in a standard 5-gram language model, i.e. word pairs more than five words apart. Long-distance agreement is a tricky issue for string-based machine translation, which is susceptible to errors such as incorrect noun/verb agreements. We show that our model is most effective for languages that are morphologically rich and allow for free word order, such as those in the Slavic family, because they contain more examples of non-local dependencies that effect fluency. See Figure 4.1 for an example of such a long-distance word agreement error.

¹<https://www.cs.jhu.edu/~ccb/publications/incremental-syntactic-language-models-for-phrase-based-translation-errata.pdf>

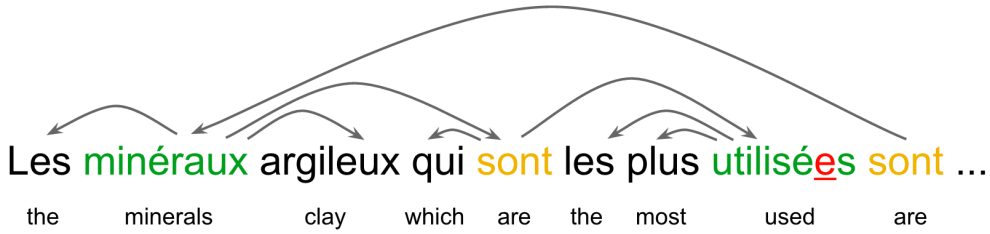


Figure 4.1: Example of long-distance agreement error in a French translation. The grammatical gender of the adjective *utilisées* (f.pl.) should be corrected to *utilisés* (m.pl.) to agree with the noun *minéraux* (m.pl.). The verb *sont* also demonstrates long-distance agreement that would be difficult to capture with a classic n -gram model.

4.2 Related Work

The idea of capturing structure in a language model has been around since the late 1990s [14], particularly in the speech recognition community. Such tree-based, or ‘structured’ language models have mainly considered only limited word histories, such as ancestors [35] or specific parent/sibling relationships [88], however more recent attempts have started to define more general syntactic word histories [91]. The beginnings of such generalized approaches can be traced back to ‘arbori-context’ trees [63], which are designed to select optimal partial histories.

Other effective syntactic approaches in recent years have included a bilingual language model [58, 70] enhanced with some dependency information [32], specifically the POS tags of parent/grandparent and closest left/right siblings, and modeling a generative dependency structure on top of a classic n -gram language model [27].

While not directly designed as ‘syntax-based’ language models, approaches based on neural networks have also been shown to be effective at capturing a more general word history than classic n -gram models. Such approaches include feed-forward [4, 85] and recurrent [62] neural network language models and more recently LSTM-based language models [96]. The most recent approach at the time of writing considers a hybrid approach of syntactic and neural network components

[86].

Two major drawbacks of neural network based approaches are that it can be difficult to ‘reverse engineer’ the syntactic knowledge learned and that model training can struggle computationally on large data.

Our approach expands on existing studies by proposing a more generalized framework for syntactic context and analyzing its effectiveness for a large range of language pairs. We show that it is applicable even to systems without target-side syntax.

4.3 Model Details

4.3.1 Classic n -grams with Linear History

The classic generative story for language models is as a Markov process. Generation of a sequence of words is based on the notion of ‘history’ or ‘context’, i.e. the ordered list of words already generated. It would be desirable to consider complete histories, however in practice this is not tractable. To counter sparsity and computational issues, a selective history must be used.

These ideas inspire the design of the classic n -gram language model. We assume an $(n - 1)$ th order Markov property and use a linear context, i.e. model the probability of any given word as being conditional on the previous $n - 1$ words. The probability of a sentence w_1, \dots, w_m can be written as:

$$P(w_1, \dots, w_m) = \prod_i^m P(w_i | w_1, \dots, w_{i-1}) \quad (4.1)$$

$$\approx \prod_i^m P(w_i | w_{i-n+1}, \dots, w_{i-1}). \quad (4.2)$$

In many cases, this linear history can be helpful in determining the next word. For example, the word ‘Francisco’ is more likely to appear after ‘San’ than ‘Los’. But when it comes to modeling other issues affecting fluency, such as word agreement, we must also use non-local context, ideally at the same time without increasing model sparsity. This is the primary motivation for our tree-based language model.

4.3.2 Syntax-Based History and t -treelets

We now consider how to define the history of a word on the syntactic level. First we define generalized tree n -grams, or ‘ t -treelets’, as the syntax-based equivalent of the classic n -gram. Our definition of t -treelets is similar to the concept of syntactic n -grams [91], which we formalize and expand over arbitrary tree structures. This is in contrast to previous work that considers only a limited subset of possible syntactic relations.

Let us assume a sentence $S = \{w_1, w_2, \dots, w_m\}$ of length m with a virtual root R and a connected tree structure $T : S \rightarrow S \cup \{R\}$ mapping each word w_i to one head $T(w_i) \in S \cup \{R\}$ with $T(w_i) \neq w_i$. The design of T can be motivated by any arbitrary set of standards, however a natural choice for machine translation applications would be dependency parses.

We now define the ‘history’ H_i for w_i as the subset of S consisting of the words visited by in-order depth-first traversal of $\{S, T\}$ starting at R and ending at w_i . The diagram on the right of Figure 4.2 shows the tree-based history of an example sentence (shown on the left).

The core reasoning behind this definition of history is that we wish to ensure that the t -treelet history of all words respects a well-defined ordering (in this case the order of visiting nodes by depth-first traversal). This ensures that we never encounter any cyclic dependencies or ambiguity when calculating the probability of an entire tree. Note that this well-defined ordering is trivial in the case of classic n -gram models, however this is an important consideration in tree-based models.

While in this study we use in-order depth-first traversal, any well-defined ordering could be used, for example to reflect the ordering used for hypothesis combination in a tree-based decoder. As an example of differences caused by this choice, an in-order depth-first traversal allows us to include the children of left-side siblings into word history, and this is useful for many word agreement problems, however this is not possible with breadth-first traversal.

Conversely we cannot make use of the right-side siblings of a word. This could be useful in rare cases such as ‘le prix fixe’ (‘the set price’) when we need to use a modifier to the right of a determiner (the adjective ‘fixe’) to determine the gender/number of its head noun (‘prix’), which in this case could be either singular

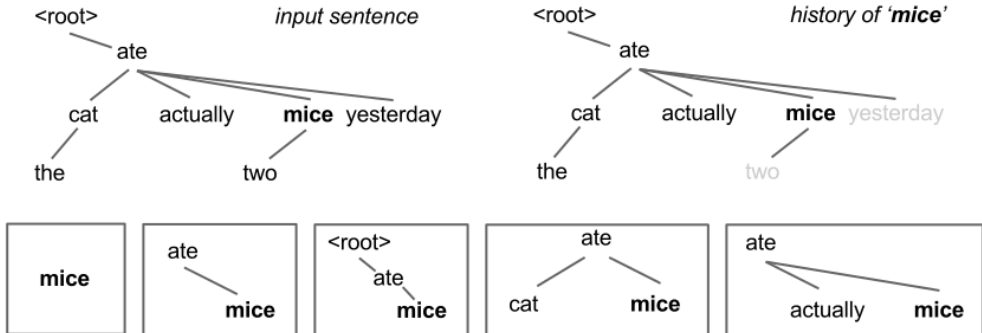


Figure 4.2: Example of t -treelet extraction. The left figure shows an example sentence $\{S, T\}$ and the right figure shows the history H for ‘mice’. The five possible t -treelets of order $l \leq 3$ are shown beneath.

or plural.

We now define the ‘ t -treelets of size l for w_i ’, as all connected subtrees $S' \in H_i$ where $w_i \in S'$ and $|S'| = l$, along with the tree structure T . See the lower half of Figure 4.2 for the t -treelets of order $l \leq 3$ extracted for the word ‘mice’ in the example sentence. Our t -treelet definition captures in particular the difference between left and right dependencies, e.g. whether w_i is to the left or right of $T(w_i)$ (we treat each case separately), and relative sibling positions, which are to our knowledge not considered in previous work.

While there is only one possible linear n -gram of given size for any word, the same cannot be said for t -treelets. Furthermore, the number of possible t -treelet shapes increases with l and depends on the sentence structure. For convenience we normalize the $\{S', T\}$ by renumbering the words and dependencies from 1 to l . This allows us to classify t -treelet shapes into groups.

The possible shapes for t -treelets for $l \leq 3$ are shown with natural language examples in Figure 4.3. For completeness we also add the empty t -treelet. There are 10 possible such t -treelets: 1 of size 1, 2 of size 2 and 7 of size 3. A major benefit of this approach is that we are able to turn each shape g ‘on’ and ‘off’, a process which is described in Section 4.3.2 below.











Type	Shape	Example	Type	Shape	Example
0		n/a	5		сделаем девочку красивой <i>make girl pretty [RU]</i>
1		le chat <i>the cat [FR]</i>	6		der Mann hat <i>the man has [DE]</i>
2		fromage bleu <i>cheese blue [FR]</i>	7		hat einen Hund <i>have a dog [DE]</i>
3		chat a mangé <i>cat has eaten [FR]</i>	8		у меня есть <i>to me exists [RU]</i>
4		il est grand <i>he is big [FR]</i>	9		قابلت طالباً جديداً <i>met student new [AR]</i>

Figure 4.3: The shapes of all possible t -treelet types for $l \leq 3$ with natural language examples and word-by-word glosses. The words marked with dark nodes represent the w_i around which the t -treelets are centered.

We can now model the probability of generating a given word w_i with t -treelets G as $P(w_i | H) \approx P(w_i | G)$. Note that the different $g \in G$ are not always independent, so this probability can be rather complicated to calculate. We found that the approximation $P(w_i | G) \approx \sqrt[t]{\prod_{g \in G} P(w_i | g)}$ works well, although it could also be possible for example to treat the individual $P(w_i | g)$ as scores and combine with a log-linear model. In our case we found that it was difficult to learn weights for a log-linear model, since in our experiments BLEU was not sensitive to changes in long-distance word agreements.

Task-Specific Shape Selection

Previous definitions of tree-based histories have considered subsets of the possible t -treelet shapes that we have defined above, such as ancestor chains $(w_i, T(w_i), \dots, T^{l-1}(w_i))$ [35] and restricted parent/sibling relations [88]. Our definition not only expands upon these, but also adds flexibility as we are able to turn each shape type ‘on’ and ‘off’ depending on the requirements of the task. An additional benefit is that it becomes possible to compare directly with previous work by simply selecting t -treelet shapes.

In particular, we found that there are types of dependency relations that may or may not affect word agreement depending on languages and parsing standards. The natural language examples shown in Figure 4.3 give classic cases where certain t -treelet shapes are important for determining word morphology. There equally are types that are never (or very rarely) used in certain languages, and we found that considering these types at times caused unnecessary noise. This is similar to using unnecessarily long n -gram sizes in classic language models, where there is often not enough gain in expressiveness to warrant the additional errors caused by sparsity and irregular smoothing.

See Section 4.5 for experimental results and a more detailed analysis of the relative performance of various special cases of our model, including comparison with previous work.

4.3.3 Smoothing

Since we conducted our translation experiments on web-scale data, we designed our model to be used with Stupid Backoff [7] smoothing, which has been shown to perform well on this kind of data.

The mathematical formulation of Stupid Backoff smoothing is shown below. The formula below is applied recursively until a known n -gram is found, and uni-gram scores are defined as $SB(w_i) = c(w_i) / \sum_w c(w)$, where $c(w)$ is the observed frequency of word w in the training corpus. The parameter α controls the degree to which we penalize backing off to shorter n -grams.

$$SB(w_i | w_{i-n+1}^{i-1}) = \begin{cases} \frac{c(w_{i-n+1}^i)}{c(w_{i-n+1}^{i-1})}, & \text{if } c(w_{i-n+1}^i) > 0 \\ \alpha SB(w_i | w_{i-n+2}^{i-1}), & \text{otherwise.} \end{cases} \quad (4.3)$$

The primary advantage of this smoothing method is that it does not require the calculation or lookup of modified t -treelet counts. This allows for fast and simple calculation of backoff probabilities, and works well when using each t -treelet type score as a feature. Note that Stupid Backoff smoothing was also used in similar work [35] that we use for comparison.

When backing off to shorter t -treelets, note that we calculate probabilities based on the shorter shape type, and that this type is always unique because of the (in-order depth-first) ordering constraint. For example (in Figure 4.3), ‘il est grand’ (type 4) is backed off to ‘est grand’ (type 2) non-ambiguously.

4.3.4 Application to SMT: Filtering and Reranking

In our experiments (see Section 4.4) we measure the translation improvement gained by reranking k -best machine translation output using our language model.

Reranking is a very flexible and simple approach. In particular we do not make any assumptions about the decoding algorithm of the underlying MT system and we are able to use a standard string-to-string system by simply parsing the output. As mentioned in the introduction, we believe that a major stumbling block for syntax-based language modeling has been the lack of applicability to string-based

MT systems (which are still the most common), and we show that for our model this is not an issue.

The obvious problem of using string output is that we cannot guarantee reliable parsing, particularly of (poorly formed) machine translation output. We propose the simple approach of using a filtering heuristic based on dependency tree consistency to reduce this problem.

We parse all k -best candidates and extract the dependency treelets of size l centered on each word that differs between each candidate and the 1-best (baseline) translation. We then discard any k -best candidates that contain any such treelets with a different dependency structure to the corresponding treelet in the 1-best translation. This simple heuristic was very effective at reducing errors caused by bad translations/parses, as simple word changes (e.g. changing the gender of a definite article) should not affect the parse tree. Naturally this filtering leads to a small reduction in recall.

4.4 Experimental Setup

We performed a series of experiments to measure the improvement in translation quality obtainable by reranking MT output using our proposed tree-based language model. In particular we were interested in improving morphological errors such as word agreement.

4.4.1 Language Choice

In our experiments we built and evaluated models for nine major languages. This allowed us to analyze clearly the types of morphological error that the proposed model was able to improve. The languages were selected from a variety of language families and all display word agreement to various degrees.

The languages chosen were: Czech and Russian [Slavic]; Hungarian [Uralic]; Dutch and German [Germanic]; French, Portuguese and Spanish [Romance]; and Hebrew [Semitic]. For consistency we used English as the source language for all translation experiments. See Table 4.1 for an overview of the characteristics of these languages affecting (long-distance) word agreements.

	Family	Cases	Genders	Other
Czech	Slavic	7	4	Highly flexible word order.
Russian	Slavic	6	3	Highly flexible word order.
Hungarian	Uralic	18 ²	0	Displays verb-object agreement.
Dutch	Germanic	0 ³	2	Diminutive inflections.
German	Germanic	4	3	Long-range word order flexibility.
French	Romance	0	2	Rich verb forms.
Portuguese	Romance	0	2	Rich verb forms.
Spanish	Romance	0	2	Rich verb forms.
Hebrew	Semitic	0	2	Highly inflected Semitic roots.

Table 4.1: Word agreement/ordering characteristics of the nine languages selected for translation experiments.

All language models were trained on mixed domain monolingual web corpora of 5–10 billion unique sentences per language.

4.4.2 Automatic Evaluation Metrics

Translation quality was measured with BLEU [74] and the language model was intrinsically evaluated using a method of evaluation we call ‘win-rate’ (see below).

The formulation of BLEU is described in Section 1.4.1. As can be seen from the definition, BLEU considers only the precision of local n -grams. BLEU has been shown in the past to be ineffective in evaluating syntax-based approaches, with improvements being ‘invisible to [such] an n -gram metric’ [86]. We also found that BLEU was unreliable at reflecting changes in translation quality for long-distance dependencies, and that the sensitivity was low because only a small fraction of words were changed by using the proposed model (for example many sentences do not contain word agreement errors). Nonetheless it was practical to use such an automatic measurement for parameter tuning.

As another point of reference, we also used a method of intrinsic language

²Sources disagree about this number, however it is generally agreed to be between 16–24.

³There remain some rare examples of declension taken from archaic usage.

		French	German	Russian
$f = 100$	BLEU	30.32	22.39	19.23
	win-rate	0.505	0.612	0.649
$f = 10$	BLEU	30.32	22.45	19.23
	win-rate	0.512	0.635	0.674
$f = 1$	BLEU	30.33	22.49	19.23
	win-rate	0.534	0.667	0.737

Table 4.2: Result of varying model size by changing t -treelet filtering threshold f in training. These results used the setting ‘AllTypes’.

model evaluation we call ‘win-rate’. For each sentence we calculated the language model score (using the proposed model) of the baseline MT system output and the reference translation. The win rate was then calculated as follows, giving the ratio of number of times our model gives a higher score to the reference translation than to the baseline output. The model can be considered useful if it can successfully give a higher score to the reference translation than the baseline MT output. While we do not claim that this metric is strongly correlated with human judgment, we believe it gives useful information and is very simple to implement.

$$\text{win-rate} = \frac{\#(\text{score}(\text{reference}) > \text{score}(\text{baseline}))}{\#\text{sentences}} \quad (4.4)$$

The classic method of intrinsic language model evaluation is perplexity, however we chose not to use this measure because it assumes normalized probabilities, which we cannot strictly guarantee when using our model approximations and Stupid Backoff smoothing.

4.4.3 Training and Lookup

Prior to model training, we tokenized the entire training corpus and collected word frequencies. Tokens with frequency less than or equal to a certain threshold (in our case 1) were replaced with an ‘unknown’ token in order to model t -treelet counts during lookup that include out-of-vocabulary tokens.

Training was conducted by parsing training sentences and counting all t -treelets of size ≤ 3 . It would be possible to use longer t -treelets however we found that there were not many cases where longer context was necessary for determining correct word agreement. To save memory t -treelets could be pruned based on frequency, however we found that this negatively impacted performance (see Table 4.2). Parsing was conducted with the shift-reduce dependency parser described in [51].

4.4.4 Reranking SMT Output

In order to evaluate the effectiveness of the proposed model we tested the ability of our language model to rerank the 1000-best translation output of a string-based SMT system.

The baseline translation system was a state-of-the-art in-house phrase-based translation system trained on large web data. The baseline used a standard 5-gram language model trained on the same data as the proposed tree-based model and for comparison also used Stupid Backoff smoothing.

The 1000-best translation candidates were filtered using the dependency tree consistency heuristic described in Section 4.3.4. We also removed noisy sentences consisting over 50% non-alphanumeric characters, and evaluated on sentences with length between 10 and 30 words.

4.5 Optimization of Model Parameters

We first explored the effects of varying our model parameters, in particular the selection of t -treelet shapes, comparing with previous work. The experiments were conducted on a development data set consisting of approximately 10000 sentences per language that were held out from our baseline and language model training data.

For comparison with previous work, we first experimented with settings enabling various sets of t -treelet shapes (for size $l \leq 3$). The setups ‘Ancestors’ and ‘Siblings’ were designed to correspond to the models of [35] and [88] respectively. Note that there are some slight differences, in particular the smoothing algorithm

for ‘Siblings’ (Shen et al. did not mention any smoothing) and the fact that our models are more general than previous work, differentiating between left and right children.

The four model variants tested were as follows:

- Ancestors: ancestors, no siblings (types: 1–2, 6–9)
- Siblings: siblings, no ancestors (types: 1–5)
- AllTypes: all t -treelet types (types: 1–9)
- Trigrams: all pure 3-grams, siblings and ancestors (types: 3–9)

4.5.1 Results

Table 4.3 shows the results for the four system variants. We can see that the most effective settings were to use all t -treelets or all trigrams, and these more general setups performed better than the more restrictive settings based on previous work. We found that using only trigrams gave better results than for all t -treelets because the lower order t -treelets often gave less reliable information (i.e. we need longer context).

Additional tuning experiments showed that improvements were made by increasing model size (reducing t -treelet filtering threshold frequency f for training, see Table 4.2). An increase of on average 0.1 BLEU per language was observed by varying the beam width from 1 to 100, and we used a beam width of 100 for all our evaluation results. We note that the parsing quality was roughly the same for all languages. For detailed parser evaluation, see [51].

We also found empirically that it was effective to penalize unseen t -treelets more heavily than in previous work [7] by changing the backoff parameter α from the standard 0.4 to 0.004 (see Section 4.3.3 for more details). We did not conduct a full-scale experiment to find the optimum value.

		Slavic		Uralic	Germanic	
System	Language	CS	RU	HU	NL	DE
Baseline	BLEU	18.75	19.19	13.60	28.42	22.40
	win-rate	-	-	-	-	-
Ancestors	BLEU	18.81	19.23	13.64	28.41	22.43
	win-rate	0.540	0.681	0.533	0.584	0.631
Siblings	BLEU	18.81	19.21	13.62	28.44	22.44
	win-rate	0.542	0.637	0.550	0.578	0.604
AllTypes	BLEU	18.83	19.23	13.63	28.44	22.44
	win-rate	0.548	0.676	0.547	0.591	0.636
Trigrams	BLEU	18.82	19.22	13.63	28.48	22.43
	win-rate	0.563	0.681	0.552	0.592	0.645

		Romance			Semitic
System	Language	FR	PT	ES	IW
Baseline	BLEU	30.25	34.51	32.36	20.96
	win-rate	-	-	-	-
Ancestors	BLEU	30.26	34.62	32.36	20.98
	win-rate	0.504	0.604	0.545	0.505
Siblings	BLEU	30.28	34.57	32.33	20.97
	win-rate	0.509	0.604	0.541	0.520
AllTypes	BLEU	30.30	34.63	32.39	20.97
	win-rate	0.509	0.615	0.539	0.517
Trigrams	BLEU	30.30	34.67	32.38	20.99
	win-rate	0.528	0.607	0.549	0.520

Table 4.3: Comparison of model formulations enabling various t -treelet types. BLEU and win-rate are shown for each proposed system. The best results are shown in bold type.

4.6 Final Evaluation and Error Analysis

4.6.1 Experimental Settings

We conducted a full evaluation of our proposed approach on nine language pairs. For the final evaluation we used the ‘Trigrams’ settings that were shown in Section 4.5 to be the most effective overall. We decided to use this setting for all language pairs, since we did not believe that the BLEU and win-rate scores gave a clear enough winner for each individual language pair.

The experiments were conducted by translating mixed-domain English web sentences that were held out from the baseline SMT system and language model training data. As we were interested in evaluating the differences between the baseline and proposed models, we translated a large test set then for evaluation randomly selected (on average) 400 sentences per language that had different output between the baseline and proposed systems. The change rates in Table 4.4 shows the percentages of sentences that were translated differently.

4.6.2 Human Evaluation

Translation quality was measured by skilled human raters in order to maximize the reliability of the evaluation. The raters were bilingual speakers of each language pair but not professional translators.

For each sentence the raters were instructed to give a score between 0 and 6 (inclusive), given the source sentence and translation, one rater per sentence, as described in Section 1.4. The rating guidelines are shown in Table 1.1 and the number of raters per language pair can be found in Table 4.4.

We calculated the following scores for each language pair:

- ‘mean-diff’: The mean difference between the sentence-level human ratings of the proposed and baseline systems.
- ‘mean-sign’: The mean difference between the number of sentence-level wins and losses (in terms of human ratings) of the proposed and baseline systems.
- ‘change-rate’: Percentage of sentences that were different between the baseline and proposed systems.

	Slavic		Uralic	Germanic	
Language	Czech	Russian	Hungarian	Dutch	German
mean-diff	+0.0771	+0.0985	+0.0600	-0.0248	+0.0599
mean-sign	+0.0771	+0.0803	+0.0600	-0.0199	+0.0599
change-rate (%)	8.51	2.74	5.57	7.04	7.57
baseline (0–6)	2.86	2.76	2.27	3.75	3.14
proposed (0–6)	2.94	2.86	2.33	3.73	3.20
Number of raters	18	61	14	25	75

	Romance			Semitic
Language	French	Portuguese	Spanish	Hebrew
mean-diff	+0.1075	-0.0050	-0.0124	+0.0099
mean-sign	+0.1025	-0.0100	-0.0299	-0.0199
change-rate (%)	4.71	10.21	4.98	6.18
baseline (0–6)	3.54	4.30	3.98	3.01
proposed (0–6)	3.65	4.29	3.97	3.02
Number of raters	74	81	73	18

Table 4.4: Human evaluation results, comparing mean difference between proposed and baseline systems, sorted by language group.

- ‘baseline’: Mean sentence-level human evaluation score for the baseline system.
- ‘proposed’: Mean sentence-level human evaluation score for the proposed system.

4.6.3 Results

Table 4.4 shows the results sorted by language family. Significantly positive ($p < 0.05$) results for ‘mean-diff’ and ‘mean-sign’ are shown in bold type. Table 4.5 shows the exact number of test sentences with each score difference (-6 to +6) between proposed and baseline systems.

	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6
Czech	0	0	0	0	3	29	307	61	1	1	0	0	0
Russian	0	0	0	0	2	18	212	35	7	0	0	0	0
Hungarian	0	0	0	0	3	29	312	53	3	0	0	0	0
Dutch	0	0	0	0	8	85	225	80	4	1	0	0	0
German	0	0	0	0	5	40	287	64	5	0	0	0	0
French	0	0	0	1	6	45	255	83	10	0	0	0	0
Portuguese	0	0	0	0	13	58	263	52	15	0	0	0	0
Spanish	1	0	1	0	9	71	250	60	5	2	0	0	3
Hebrew	0	0	0	1	8	48	297	38	5	2	3	1	0

Table 4.5: Number of test sentences with each score difference (-6 to +6) between proposed and baseline systems.

The results show a significantly positive⁴ improvement for Czech, Russian, Hungarian, German and French, with more fluent output as judged by human raters. In particular, all the languages displaying noun declension were significantly improved (Czech, Russian, Hungarian and German). The translation quality for Romance languages (with the exception of French), Hebrew and Dutch did not change significantly when using a tree-based language model. In the next section we analyze these findings in detail.

4.6.4 Discussion

The results of the human evaluation showed a noticeable difference between the effectiveness of the tree-based language model for different languages. In particular, the morphological characteristics of each language (to some extent captured by the language family) appear to effect greatly the utility of a structured language model in comparison to the baseline, which uses a standard n -gram model.

All nine languages require correct word agreement for high fluency, however

⁴More precisely, we calculated the 95% confidence interval for the observed mean using the Student's t -distribution, where the degrees of freedom were set to the sample size minus 1. The result was deemed significantly positive if the lower bound was greater than zero.

the type and nature of these agreements varies between language. For example, adjective-noun and noun-verb agreement in Romance languages can be expressed relatively simply with an n -gram model as the words in question normally appear together, which could explain why we saw less improvement for this language family. In contrast, case choice in Slavic languages requires consideration of complicated and long-distance dependencies, which is consistent with the large improvement shown by our proposed system. Similar observations that such approaches are more effective for languages with relatively free word order have been made in previous work [86].

The magnitude of improvement per language also showed some correlation with the baseline translation quality. Analysis of the scores given by human raters showed a tendency for lowered sensitivity to word endings when the translation quality was high. We found many examples of generally well-translated sentences (particularly in Portuguese and Spanish) that were given a score of 6 (out of 6) irrespective of word ending errors. This means that there were a number of improvements not reflected in our results. Conversely, Russian sentences with a lower average score tended to gain or lose a whole point when word endings were changed. This could also be due to the type of errors themselves, as for example adjective agreement mistakes are unlikely to impede understanding as much as case errors, and shows the importance of accurate long-distance agreement.

4.6.5 Error Categorization

Table 4.6 gives an error categorization for a random sample of ten incorrect sentences for each of five languages (Dutch, French, German, Portuguese and Russian). The categories are defined as follows:

- OOV (26%): t -treelet not seen in training data
- Meaning (22%): Meaning of original sentence changed (e.g. tense)
- Tricky (16%): Various tricky cases⁵

⁵For example, incorrect choice of indicative/subjunctive, case selection requiring semantic inference.

	Dutch	French	German	Portuguese	Russian	Total
OOV	0	0	5	4	4	26%
Meaning	3	3	1	4	0	22%
Tricky	3	0	2	0	3	16%
Parse	1	2	1	0	3	14%
Noise	2	2	0	2	0	12%
Context	1	3	1	0	0	10%

Table 4.6: Error categories for analyzed test sentences.

- Parse (14%): Parse error caused incorrect t -treelet lookup
- Noise (12%): Broken input sentence, incorrect human rating
- Context (10%): Model used inappropriate context (e.g. too short)

Overall the most common cause for errors was out-of-vocabulary t -treelets, particularly phrases involving rare nouns. Our current model does not attempt to guess deep information, such as the gender or case of out-of-vocabulary words, however this would make for interesting future work. The second most common error was changing the original sentence meaning by for example modifying the tense or changing singular to plural. This could be improved in the future by considering a bilingual approach incorporating source tokens.

Despite parse error reduction by our filtering method, errors in the parsing of the training corpus still led to learning some incorrect t -treelets. In particular, structures such as ‘the JJ NN and NN’ (for example ‘the unusual character and Amy’) caused the most errors, as they were often incorrectly parsed and caused word agreement errors (e.g. ‘unusual’ had plural agreement). While in the majority of cases the proposed model formulation picked sufficient and appropriate context, there were some difficult cases requiring larger context that was not covered by length 3 t -treelets, in particular for words with many (> 3) siblings.

4.7 Example Sentences (Improved)

Below we give illustrative examples of improved translations for the five languages analyzed in Table 4.6 (Dutch, French, German, Portuguese and Russian).

Dutch

- **Input:** Then you might also be interested in Innolog Holdings Corporation
- **Baseline:** Dan **is dit** misschien ook geïnteresseerd in Innolog Holdings Corporation
- **Proposed:** Dan **ben jij** misschien ook geïnteresseerd in Innolog Holdings Corporation

French

- **Input:** This is an example of how a complex data structure must be broken in basic data elements.
- **Baseline:** Ceci est un exemple de la manière dont une structure de données complexe doit être **brisé** en éléments de base de données.
- **Proposed:** Ceci est un exemple de la manière dont une structure de données complexe doit être **brisée** en éléments de base de données.

German

- **Input:** The aim of this game is to help your child with their shape and visual recognition skills.
- **Baseline:** Das Ziel dieses Spiels ist es, **Ihr** Kind mit ihrer Form und visuelle Erkennung Fähigkeiten helfen.
- **Proposed:** Das Ziel dieses Spiels ist es, **Ihrem** Kind mit ihrer Form und visuelle Erkennung Fähigkeiten helfen.

Portuguese

- **Input:** Many of the students play video games all the time at home.
- **Baseline:** Muitos dos estudantes **jogar** videogames o tempo todo em casa.
- **Proposed:** Muitos dos estudantes **jogam** videogames o tempo todo em casa.

Russian

- **Input:** In 2013 we are believing for revival over your church and our cities.
- **Baseline:** В 2013 году мы верим в возрождение над вашей церкви и наших **городах**.
- **Proposed:** В 2013 году мы верим в возрождение над вашей церкви и наших **городов**.

4.8 Example Sentences (Worsened)

Below we give illustrative examples of worsened translations.

Dutch

This error was caused by incorrect parsing of the noun sequence of repeated ‘wijn’ (wine) tokens.

- **Input:** Donnafugata Wines - White wines, Red wines, Natural sweet wines
- **Baseline:** Donnafugata **Wijnen** - Witte wijn, rode wijn, natuurlijke zoete wijnen
- **Proposed:** Donnafugata **Wijn** - Witte wijn, rode wijn, natuurlijke zoete wijnen

French

While the change is grammatically correct, it incorrectly modifies the original tense (past into present).

- **Input:** Aurel Stein’s identification seemed rather tentative
- **Baseline:** L’identification de Aurel Stein **semblait** plutôt provisoire
- **Proposed:** L’identification de Aurel Stein **semble** plutôt provisoire

German

The error is caused by an OOV *t*-treelet ‘dem taktischen Rückzug’ (‘the tactical withdrawal’).

- **Input:** The song is sung by a soldier on the island of Corfu, following the tactical withdrawal of the Serbian Army through east and west Albania.
- **Baseline:** Der Song wird von einem Soldaten auf der Insel Korfu gesungen, nach **dem** taktischen Rückzug der serbischen Armee durch Ost und West Albanien.
- **Proposed:** Der Song wird von einem Soldaten auf der Insel Korfu gesungen, nach **der** taktischen Rückzug der serbischen Armee durch Ost und West Albanien.

Portuguese

This is caused by a parse error around the infinitive construction ‘não saber’ (‘not knowing’).

- **Input:** Charms, demons in caves, and the mediator’s art of not knowing
- **Baseline:** Pendentes, demônios em cavernas, e arte do mediador de não **saber**
- **Proposed:** Pendentes, demônios em cavernas, e arte do mediador de não **sabendo**

Russian

This is a tricky example where the Russian preposition ‘на’ (‘to/on’) can take either of two grammatical cases (accusative or prepositional) depending on meaning, and the incorrect meaning is selected.

- **Input:** Sending a notification to update a badge on the tile
- **Baseline:** Отправка уведомления обновить значок на **плитке**
- **Proposed:** Отправка уведомления обновить значок на **плитку**

4.9 Summary

In this chapter we have described a generalized dependency tree language model for machine translation. We performed a thorough human evaluation on nine major languages using models trained on large web data, and have shown significantly positive improvement in translation quality for five morphologically rich languages.

Analysis suggests that a generalized tree-based language model is best suited to languages groups such as Slavic and Uralic that display many non-local features such as cases, as we saw no significant improvement over a classic n -gram language model for groups such as Romance languages with high baseline quality and few non-local word agreements. Despite the common concern that tree-based language models are incompatible with string-based MT systems, we have shown that our model is capable of performing well even in this scenario by using filtered parses of string MT output.

As future work we would like to experiment with other methods of integrating the language model score into machine translation systems. The natural starting point is to query the tree language model during decoding, as the reranking method proposed in this paper has access to a limited number of hypotheses and does not integrate other features that are available to the decoder. In addition, as we have shown that BLEU is insensitive to changes made using a syntax-based language model, in the future we would like to explore metrics based on syntactic n -grams [86]. This would allow for improved model tuning.

We have shown in this chapter that dependency tree language models can be effective in improving translation fluency for morphologically rich languages, however there remain two major drawbacks. The first is our assumption of the availability of a high-quality target-side parser. Furthermore, we require particularly robust parsing for reranking (potentially malformed) string output. The second disadvantage is that we do not consider source-side context, which can be important for preserving the intended meaning of a translation. We address these two issues in the next chapter.

Chapter 5

Dependency Tree Language Model II

In the previous chapter we learned that dependency tree language models can be effective in improving word agreement for morphologically rich languages. Our proposed approach relied however on high quality target-side parsing and did not consider source-side context.

In this chapter we present a practical solution to syntax-based post-editing that does not require target-side parsers. We exploit source-side dependencies from a resource-rich language, such as English, to build a projected syntax language model that is independent of target language. To overcome sparsity issues caused by mapping syntax across distant and low-resource language pairs, we design simple yet inclusive context rules.

The proposed approach gives a significant improvement in translation quality, judged by skilled human raters, for all 20 languages tested. Our post-editing method is simple, fast and decoder independent. To improve the coverage of our approach we also consider an extension to a more general delexicalized language model.

5.1 Introduction

Word agreement is a bottleneck in the improvement of automatic translation. Morphologically rich languages have a large number of surface word forms and this leads to data sparsity problems that affect both phrase-based and neural network MT systems. The problem is particularly challenging for morphologically poor-to-rich language pairs, such as English–Russian, where it can be difficult to infer target-side features such as grammatical cases from the source language.

Previous work has used a variety of source-side approaches such as source enrichment with generated morphology [2, 100] and generation of synthetic phrases [12]. The most effective target-side approaches have involved dependency-based language models [88, 35, 80], however these suffer from unreliable parsing accuracy. Bilingual language models are another possibility [32], however they suffer from sparsity problems.

The main idea of this study is to project source-side syntax (specifically dependency trees) to the target-side using word alignment¹, then to construct a target-side language model using these projected dependencies.

There are a number of benefits of projecting the source-side syntax. The first is that we do not require a parser for the target-side, which is often not available (or of low quality) for morphologically rich languages. In most cases the source language will be English, for which there exist high-quality dependency parsers.

The second major advantage of our approach compared to the use of target-side language models is that source-side information is not distorted by the noisy translation process. It is difficult to use a target-side language model for post-editing because we must either use an X-to-tree decoder or attempt to parse decoder output that is likely to contain errors. In contrast we can easily employ projected language models for post-editing.

The greatest potential fallback of a mapping approach is that we assume the syntax of the source language is sufficiently similar to the target language. While

¹The generation of word alignments is out of the scope of this study. Our approach simply requires each source word to have a corresponding target word (or null token), which can be inferred using traditional SMT word alignment, dictionary-based matching or extracted from a neural attention model.

we cannot assure this for long sentence fragments, for most context affecting word agreement (such as verb–noun and noun–adjective pairs) we only require bigram mappings.

We elect to use post-editing as it is simple, efficient and decoder-independent. Furthermore, we edit the 1-best translation directly as this is k times more efficient than popular reranking approaches that process k -best lists. Unlike much post-editing work, such as the WMT15 post-editing shared task [6], we do not require human post-edit logs as training data, and instead train directly on the parallel corpus.

5.2 Syntactic Transfer

5.2.1 Related Work

Syntactic transfer by projection has been proposed for a variety of NLP tasks, such as training POS taggers, named-entity taggers and parsers [106, 43]. The basic principle is to transfer knowledge from a resource-rich language to a resource-poor language.

In the context of machine translation, syntax projection has mainly been used to enrich training data for tree-based MT systems, i.e. parse trees of one or both sides of a parallel corpus. For example, dependency parsers can be trained by projecting an English treebank to a resource-poor language [95]. It has been shown that MT systems trained with such projected parses can perform as well as those trained on standard treebanks [45].

Syntactic transfer can be used not only for creating training data for syntax-based MT systems, but also indirectly to train a reordering model for phrase-based systems [33], in this case inferring source syntax from the target side. As another example, the parser with higher quality in a tree-to-tree scenario can be used to improve the parse trees generated by the lower quality parser [90].

To the best of our knowledge, this is the first study to apply syntactic transfer to structured language modeling and MT post-editing.

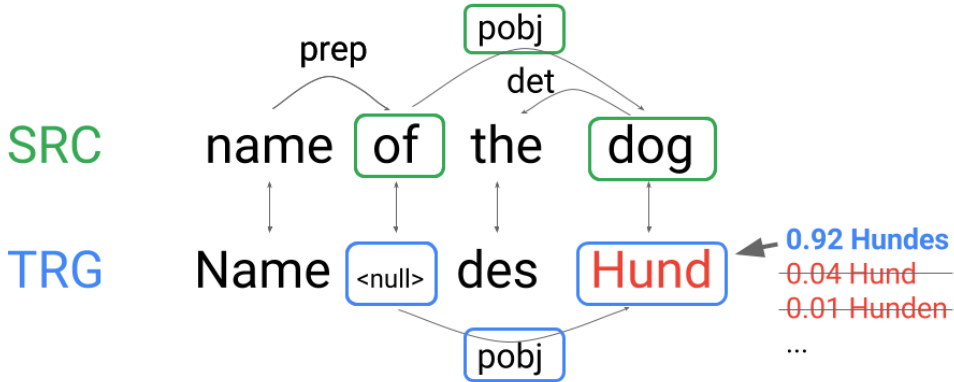


Figure 5.1: Post-editing an incorrect word form. Firstly the context is extracted for the word in question (‘Hund’) on both source (green) and target (blue) sides using mapping based on the word alignment (shown with double-headed arrows). ‘Hund’ is replaced by ‘Hundes’, the word with the highest probability of having the marked bilingual context according to the projected dependency language model.

5.2.2 Source-to-Target Dependency Mapping

In the following discussion we map source dependency trees to the target side as follows. Consider a source-side sentence (ordered multiset of words) $S = \{s_1, \dots, s_m\}$ and target-side sentence $T = \{t_1, \dots, t_n\}$ with artificial root nodes s^* and t^* . Let source-side dependencies be denoted by $d: S \times (S \cup \{s^*\}) \rightarrow L$ for dependency labels L (including a null label l_0) and source-to-target word alignment be $a: S \rightarrow T \cup \{t_0\}$, where t_0 is a null token. The projected dependency map d' (on the target-side) is then defined by $d': T \times (T \cup \{t^*\}) \rightarrow L$ where:

$$a(s_i), a(s_j) \mapsto \begin{cases} d(s_i, s_j), & \text{if } a(s_i) \neq a(s_j) \neq t_0 \\ l_0, & \text{otherwise} \end{cases} \quad (5.1)$$

Figure 5.1 shows an example of dependency projection for the *pobj* relation between the source-side noun ‘dog’ and its head ‘of’. ‘Dog’ is mapped to ‘Hund’ and ‘of’ to the target-side null token via word alignment (shown by double-ended arrows), and the dependency is copied to the target side with its label *pobj*.

5.3 Projected Language Model

5.3.1 Word Context

Previous work has considered a number of definitions of word context for dependency trees. These include parent/siblings [88], ancestors [35] and general syntactic n -grams [80]. While larger, more general contexts are effective in the case of standard target-side language models, there are two major disadvantages: sparsity and projection difficulty.

Data sparsity is problematic for morphologically-rich languages and much of the context used in previous work is irrelevant to improving word agreement. We therefore design simple rules to capture the important linguistic clues (for example the modified noun for determining adjective endings) as opposed to considering arbitrary neighboring words. These rules are sufficient to cover the large majority of word agreements for 20 major languages (see Section 5.4.2).

The projected syntax setting requires us to map dependencies for large source subtrees to the target side, which can be difficult (even for humans) for distant language pairs. By focusing on the important words, our approach does not require high quality word alignment and reduces sparsity, allowing us to use not only monolingual but bilingual tree contexts.

We define three types of word context for ‘noun-like’, ‘adjective-like’ and ‘verb-like’ structures, shown in Figure 5.2. ‘Noun-like’ captures noun inflections, such as for number, gender and case, when used as direct/indirect objects or in prepositional phrases. ‘Adjective-like’ captures modifier inflections based on the modified noun and its grammatical case. ‘Verb-like’ covers verb conjugations based on the number/person/gender of their subject. We also include any auxiliary modifiers (such as ‘will’ and ‘let’) to cover inflections for modal constructions.

The context type of a word is determined by the dependency label² between the word and its head, as shown below. We ignore words with other labels, such as conjunctions and punctuation.

- Noun-like: *dobj*, *pobj*, *iobj*

²We use the Stanford dependency labels:

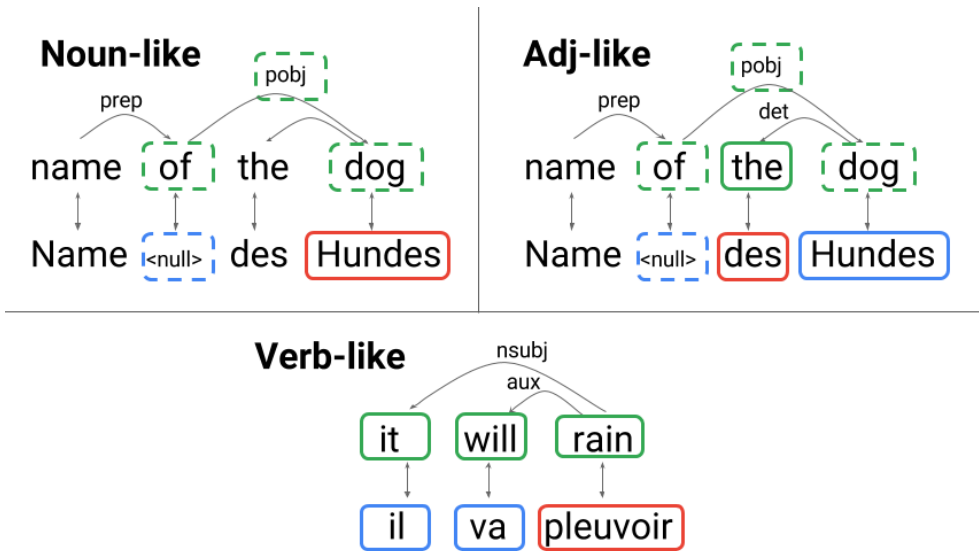


Figure 5.2: Definition of the three types of context: ‘Noun-like’, ‘Adjective-like’ and ‘Verb-like’. The green and blue boxes denote respectively the source and target context for the word marked in red. Dashed lines indicate optional context for languages displaying case agreement.

- Adjective-like: *amod*, *det*, *predet*, *poss*
- Verb-like: *nsubj*

5.3.2 Exploiting Linguistic Knowledge

While our rules are language-independent, they can be easily augmented with language-dependent knowledge.

We found that case endings caused the majority of morphological errors for languages such as Russian, however there were no such errors for French, which does not inflect nouns for case. Since we know in advance whether the target language has grammatical cases, we are able to decide whether to include richer context or to ignore certain agreements (e.g. between prepositions and nouns) in order to reduce model sparsity and undesired edits. The dashed boxes in Figure 5.2 show the words we ignored for caseless languages.

Another common phenomenon we found was the necessity to consider definiteness of nouns in some Germanic (particularly Scandinavian) languages. For example, the English ‘a green house’ is translated into Norwegian as ‘et grønt hus’, however ‘the green house’ should inflect to ‘det grønne huset’, with both the noun and adjectives changing to show definiteness. We therefore included all dependencies with the label ‘det’ to noun-like and adjective-like contexts.

5.3.3 Language Model Training

A language model can be trained to predict the most likely word given some bilingual context. We count the words appearing with each bilingual context in a parallel corpus and store them as key-value pairs, where the key is the context and the value is a list of the possible words W with their corresponding frequency across the parallel corpus. The model score for each word w given a context tree c is given as:

$$P(w | c) = \frac{f_{w,c}}{f_c} \quad (5.2)$$

where f_c is the corpus-wide frequency of context c and $f_{w,c}$ is the number of occurrences of w with context c .

For an example, see Figure 5.1, which shows the possible words W (‘Hundes’, ‘Hund’, ...) for a bilingual context c (green and blue words) and their language model scores $P(w | c)$.

5.3.4 Post-editing

The following algorithm is then used to post-edit translations of a baseline system using the projected language model. Firstly we parse the input sentence and map the source-side parse tree to the target-side baseline output. For each word in the translation, we first check the dependency label to its head and select the rule type defined in Section 5.3.1 (or skip if there is no matching rule). The context c for that rule type is extracted and the most likely word w' given the context is returned as follows:

$$w' = \underset{w}{\operatorname{argmax}} P(w | c) \quad (5.3)$$

The process is repeated for each word in the sentence in left-to-right order. While we also tried using a depth-first traversal of the projected target-side dependency tree to allow for multiple edits in a natural hierarchy, this did not give any significant improvement over the simplest approach. The same was true for a lattice search over all post-editing candidates, obtaining only a small improvement for the considerable increase in computational complexity. We therefore used the left-to-right one-pass approach, whose time complexity is linear in the sentence length and does not require generation of a k -best list.

To improve the post-editing accuracy we used a maximum entropy classifier [5] to decide whether to commit each edit based on a variety of features θ . The decision to edit was made if and only if $w \cdot \log \theta > b$, where w is a weight vector and b is a bias term. The weights and bias were tuned to maximize human evaluation scores (see Section 5.4.2) on a development set for the post-edited system versus the baseline.

In our experiments the features used were: projected tree LM score, baseline LM score (standard n -gram language model), sentence length, context type (3 binary features) and additional context type (2 binary features).

5.4 Experiments

We performed comprehensive experiments to assess the improvement in translation quality obtained by post-editing the output of a strong baseline system, which we treated as a black box. Translation experiments were performed for 20 target languages with English used as the source language, and the baseline and post-edited translations were evaluated by human raters (see Section 5.4.2).

The baseline was a state-of-the-art in-house MT system trained on a large-scale web corpus. The baseline combined SMT and NMT approaches and used settings optimized for the best baseline translation quality for each language pair. We used a standard n -gram language model trained on the target side of the parallel corpus.

The proposed (projected syntax) language model was trained on the same parallel corpus as the baseline. We used the same word alignments as in the

baseline MT system and the source (English) text was parsed with a shift-reduce dependency parser similar to previous work [51]. Evaluation was conducted on 400 randomly held-out web sentences for each language pair, and we used another 400 sentences to tune the post-edit classifier.

We used the following languages: Bulgarian (bg), Catalan (ca), Czech (cs), Danish (da), German (de), Greek (el), Spanish (es), French (fr), Croatian (hg), Italian (it), Lithuanian (lt), Latvian (lv), Dutch (nl), Norwegian (no), Polish (pl), Portuguese (pt), Russian (ru), Slovenian (sl), Swedish (sv), Ukrainian (uk). As for the additional context features explained in Section 5.3.2, we used grammatical cases for Croatian, Czech, German, Greek, Hungarian, Latvian, Lithuanian, Polish, Russian, Slovenian and Ukrainian; and we marked definiteness for Danish, Norwegian and Swedish.

5.4.1 Filtering

Since we were working with a noisy web corpus we applied some simple filtering to reduce model error. We removed any word/context pairs from the training data that appeared fewer than 5 times in total and kept only the words that were seen in at least 10% of examples for each context.

To ensure we only edited words with morphological variants, we filtered any candidates whose word endings differed by more than 3 characters, for example replacing ‘cat’ with ‘cats’ but not ‘crabs’. We filtered tokens containing multiple punctuation marks and rare short words of fewer than 5 characters.

5.4.2 Evaluation

Similar to our evaluation in Chapter 4, translation quality was measured by skilled human raters in order to maximize the reliability of the evaluation. Raters were instructed to give a score between 0 and 6 (inclusive) for each of the baseline and proposed system translations, and the rating guidelines are shown in Table 1.1.

We calculated the following scores for each language pair:

- ‘CR’: Change rate, i.e. percentage of sentences that differed between the baseline and proposed systems.

- ‘mean-diff’: The mean difference between the sentence-level human ratings of the proposed and baseline systems.
- ‘mean-sign’: The mean difference between the number of sentence-level wins and losses (in terms of human ratings) of the proposed and baseline systems.

5.4.3 Results and Error Analysis

The results are shown in Table 5.1. The projected syntax post-editing model was able to improve significantly the translations for all 20 of the target languages tested compared to a competitive baseline³.

We performed error categorization for 20 random sentences for each of three languages (German, French and Russian) to analyze the main causes of incorrect post-edits. The categorization is shown in Table 5.2 and the meaning of each category is explained below:

- Noise: Broken input sentence, meaningless baseline translation, bad human evaluation
- Tricky: Error caused by a special/rare case⁴
- Model: Incorrect word form selected by model (training noise)
- Mapping: Incorrect mapping caused by incompatible source and target structures⁵
- Meaning: Grammatically correct but changing sentence meaning (e.g. replacing ‘he’ with ‘she’)
- Parse: Incorrect source-side parsing
- Class: Original word not replaced by a morphological variant

³We omit the details of the baseline in this review copy for anonymity.

⁴For example, Russian numbers that take the genitive case, verbs in the subjunctive or imperative mood, word-sense disambiguation errors.

⁵For example, ‘the entire X’ on the source side becoming ‘the entirety of X’ on the target side.

	CR	mean-diff	mean-sign
bg	0.63	0.34 [0.25, 0.42]	0.28 [0.21, 0.35]
ca	0.76	0.15 [0.08, 0.22]	0.14 [0.08, 0.20]
cs	4.34	0.12 [0.05, 0.18]	0.10 [0.04, 0.17]
da	1.70	0.45 [0.38, 0.52]	0.42 [0.35, 0.48]
de	2.82	0.29 [0.22, 0.37]	0.24 [0.19, 0.30]
el	1.10	0.07 [0.01, 0.13]	0.06 [0.01, 0.11]
es	1.09	0.27 [0.18, 0.37]	0.23 [0.17, 0.30]
fr	1.51	0.10 [0.02, 0.17]	0.10 [0.04, 0.16]
hr	2.15	0.19 [0.12, 0.25]	0.17 [0.11, 0.23]
it	1.07	0.27 [0.17, 0.36]	0.21 [0.15, 0.27]
lt	3.72	0.24 [0.18, 0.31]	0.21 [0.15, 0.27]
lv	3.76	0.11 [0.06, 0.17]	0.11 [0.06, 0.16]
nl	2.58	0.46 [0.38, 0.54]	0.42 [0.34, 0.49]
no	1.34	0.23 [0.16, 0.31]	0.22 [0.16, 0.29]
pl	2.97	0.11 [0.03, 0.19]	0.11 [0.04, 0.18]
pt	2.02	0.29 [0.20, 0.39]	0.24 [0.18, 0.31]
ru	2.73	0.09 [0.02, 0.15]	0.06 [0.01, 0.12]
sl	3.42	0.10 [0.05, 0.15]	0.08 [0.04, 0.13]
sv	1.70	0.34 [0.26, 0.42]	0.28 [0.22, 0.35]
uk	2.38	0.07 [0.01, 0.14]	0.06 [0.01, 0.11]

Table 5.1: Mean sentence-level improvement compared to baseline with confidence interval for $p < 0.05$. All results are significantly positive.

	DE	FR	RU	Total
Noise	7	7	2	16
Tricky	0	5	8	13
Model	5	1	5	11
Mapping	4	3	3	10
Meaning	2	3	0	5
Parse	1	1	2	4
Class	1	0	0	1

Table 5.2: Error categorization for sentences worsened by the proposed post-editing method.

	DE	FR	RU	Total
OOV	4	2	5	11
Context	2	5	0	7
Tricky	1	2	3	6
Noise	2	0	1	3
Parse	1	0	1	2
Filtered	0	1	0	1

Table 5.3: Categorization of baseline errors not improved by the proposed system.

Among the real losses (ignoring errors caused by data/evaluation noise), the clearest problem was handling tricky cases where isolated contexts were insufficient to determine word endings. While it is possible to reduce such errors by enlarging context size, we found that doing so caused more negative side effects by increasing model sparsity than it did to fix these errors.

5.5 Improving Coverage with Delexicalization

Our experiments showed a clear increase in translation quality for all languages tested. However the coverage was not as high as we expected.

We performed analysis of our baseline system and found that on average there

	Wiktionary		Lexicalized		Wiktionary + Lexicalized	
	unigram	full context	unigram	full context	unigram	full context
CS	10.6	6.3	98.2	34.8	98.2	36.3
DE	27.3	16.9	96.9	61.4	96.9	64.7
ES	63.5	63.2	98.6	80.3	98.6	85.7
FR	65.0	64.6	98.5	77.9	98.5	84.5
PT	43.4	43.4	98.3	77.5	98.3	82.0
RU	42.7	23.2	96.1	51.9	96.1	57.8

Table 5.4: Coverage of unigrams and full contexts using Wiktionary, our lexicalized model and delexicalized model. Coverage is defined as the percentage of LM lookups finding matches at inference time.

	Lexicalized		Delexicalized	
	CR	mean-diff	CR	mean-diff
CS	4.42	0.119 [0.058, 0.181]	4.44	0.204 [0.128, 0.281]
DE	2.91	0.280 [0.185, 0.376]	5.06	0.197 [0.117, 0.277]
ES	1.06	0.191 [0.061, 0.321]	1.30	0.310 [0.178, 0.441]
FR	1.42	0.289 [0.168, 0.410]	1.91	0.144 [0.041, 0.247]
PT	1.84	0.374 [0.243, 0.505]	2.58	0.349 [0.223, 0.475]
RU	2.81	0.122 [0.024, 0.219]	1.71	0.187 [0.036, 0.337]

Table 5.5: Human evaluation results of delexicalized model with original lexicalized model results for comparison. The coverage is greatly improved for similar translation quality.

were morphological errors in roughly 3–5% of sentences for Romance and Germanic languages and up to 25% of sentences for Slavic languages displaying rich inflections. The proposed model was not able to correct all of these errors, with a relatively low change rate for complex and low resource languages.

5.5.1 Change Rate Analysis

We analyzed the reasons why incorrect word forms were not improved and summarized the findings in Table 5.3. The error categories are the same as in Table 5.2 with the addition of ‘OOV’ (out-of-vocabulary words and unseen n -grams), ‘Context’ (context size too small to find error⁶) and ‘Filtered’ (filtered by the rules described in Section 5.4.1).

5.5.2 Delexicalization

The most common cause of drop in change rate was OOV words and unseen n -grams, in particular for Russian and German, which display rich case endings and agglutination respectively. We observed that many errors were caused by unseen combinations of known words. For example, we were able to correct agreements such as ‘vin blanche’ (white wine) but not ‘vin verte’ (green wine), as we had not seen the term ‘green wine’ even though the individual words are common. The word ‘vin’ is not necessary however to determine the correct adjective form, instead only the information that it is a singular masculine noun. Based on this observation we attempted to generalize the proposed language model with delexicalization.

We mined word inflections from Wiktionary⁷ for six languages (Czech, German, Spanish, French, Portuguese and Russian), creating a mapping of word forms to grammatical attributes including gender, number and person. All available words and attributes (as of early 2016) were included and we set a special wildcard value to all unknown attributes.

The language model was retrained, replacing all context words with their corresponding list of tags from the mined lexicon. In the example above of ‘vin verte’,

⁶For example, not finding the incorrect agreement between ‘she’ and ‘were’ in the sentence ‘She swam in the sea and *were* bitten by a shark.’

⁷<https://en.wiktionary.org>

we replaced the context ‘vin’ with *singular-masculine*. Whenever a word was not found in the lexicon we backed off to the original (lexicalized) form.

Table 5.4 shows the increase in unigram and full context coverage we were able to achieve by delexicalizing the model. Despite the Wiktionary coverage varying considerably by language, the proposed model is able to improve full context coverage for all languages tested.

We conducted a translation experiment with the delexicalized language model, using the exact same settings as the experiment in Section 5.4 with the lexicalized model. Table 5.5 shows the results, with the lexicalized model scores for comparison. The results show a considerable improvement in change rate for all languages except Russian⁸ at on average a similar level of translation quality.

We conclude that model coverage (and depending on language also quality) can be greatly improved by delexicalization. This method is only possible however for languages for which we have a sufficiently large inflection lexicon.

5.6 Summary

In this chapter we have demonstrated that the quality of target-side word agreement can be improved significantly using mapped word dependencies. This allows us to use a dependency tree language model for languages without dependency parsers. We proposed a simple definition of linguistically motivated context that was powerful enough to show improvement for all 20 languages tested. Our post-editing approach is simple and fast, requiring no dependence on decoding algorithm.

Our extension was able to improve OOV n -gram coverage however there remains room for improvement in recall. We would like to combine morphological form generation, for example using character-based neural networks [29], with our delexicalized approach to further mitigate the data sparseness problem. While the aim of our study was to show improvement in a decoder-independent setting, it would also be interesting to experiment whether integration into a tree-to-tree

⁸A side effect of delexicalization is increase in filtering, which causes an increase in translation quality at the expense of coverage.

decoder can further improve translation quality.

This chapter concludes our exploration of dependency tree language models. We have found that the keys to effectual language modeling lie in the use of source-side context and exploitation of accurate parsing on both source and target sides whenever available.

Chapter 6

Dependency T2T Reordering

We now turn our attention to the issue of reordering, which was identified to be the most prevalent case of errors in our analysis in Section 2.6.3. We ask how target-side dependency syntax can be used to improve reordering for tree-to-tree translation systems.

Syntax-based reordering is relatively simple for binarized constituency parses, as it is sufficient to decide either to retain the order of children or to swap them. However, reordering is considerably tougher for dependency grammar, as words can have arbitrarily many children, giving $n!$ possible reorderings for n children. It can also be desirable to consider subtrees not covering all children for building translation rules, and this leads to further complications. Previous approaches have tackled these problem by restricting the definition of grammar rules, reducing the expressive power of the translation model.

In this chapter, we propose a generalized model for dependency tree-to-tree reordering based on flexible non-terminals that can compactly encode multiple insertion positions. We explore how insertion positions can be selected even in cases where rules do not entirely cover the children of input sentence words. The proposed method greatly improves the flexibility of translation rules at the cost of only a 30% increase in decoding time, and we demonstrate a 1.2–1.9 BLEU improvement over the system described in Chapter 2.

6.1 Introduction

Translation is most commonly performed by splitting an input sentence into manageable parts, translating these segments, then arranging them in an appropriate order. The first two steps have roughly the same difficulty for close and distant language pairs, however the reordering step is considerably more challenging for language pairs with dissimilar syntax. We need to be able to make linguistic generalizations, such as learning to translate between SVO and SOV clauses and converting post-modifying prepositional and pre-modifying postpositional phrases [78]. Such generalizations often require syntactically motivated long-distance reordering.

The first approaches to reordering were based on linear distortion [50], which models the probability of swapping pairs of phrases over some given distance. The linear distance is the only parameter, ignoring any contextual information, however this model has been shown to work well for string-to-string translation. Linear reordering was improved with lexical distortion [99], which characterizes reordering in terms of type (monotone, swap, or discontinuous) as opposed to distance. This approach however is prone to sparsity problems, in particular for distant language pairs.

In order to improve upon linear string-based approaches, syntax-based approaches have also been proposed. Tree-to-string translation has been the most popular syntax-based paradigm in recent years, which is reflected by a number of reordering approaches considering source-only syntax [54, 68]. One particularly interesting approach is to project source dependency parses to the target side and then learn a probability model for reordering children using features such as source and target head words [78].

While tree-to-tree translation [34, 23, 19] has been somewhat less popular than tree-to-string translation, we believe there are many benefits of considering target-side syntax. In particular, reordering can be defined naturally with non-terminals in the target-side grammar. This is relatively simple when the target structure of rules is restricted to ‘well-formed’ dependencies [88], however in this study we consider more general rules with flexible non-terminal insertion positions.

Source	Target	Rule Extracted

Figure 6.1: Examples of tree-to-tree translation rules extracted from an aligned and parsed bitext. Colored boxes represent aligned phrases and [X] is a non-terminal.

6.2 Dependency Tree-To-Tree Translation

Dependency tree-to-tree translation begins with the extraction of translation rules from a bilingual corpus that has been parsed and word aligned. Figure 6.1 shows an example of three rules that can be extracted from aligned and parsed sentence pairs. In this study we consider rules similar to KyotoEBMT ‘examples’ described in Section 2.4.2.

The simplest type of rule, containing only terminal symbols, can be extracted trivially from aligned subtrees (see rules 2 and 3 in Figure 6.1). Non-terminals can be added to rules (see rule 1 in Figure 6.1) by omitting aligned subtrees and replacing on each side with non-terminal symbols. We can naturally express phrase reordering as the source/target-side non-terminals are aligned.

Decoding is performed by combining these rules to form a complete translation, as shown in Figure 6.2. We are able to translate part of the sentence with non-ambiguous reordering (‘read a magazine’), as we can insert ‘雑誌 → a magazine’

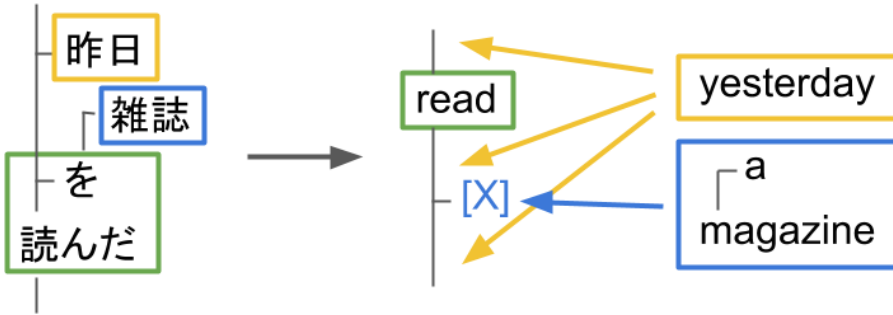


Figure 6.2: Combination of translation rules, demonstrating non-terminal substitution and multiple possible insertion positions for a non-matching input phrase (‘昨日’).

into the rule ‘[X] を 読んだ \rightarrow read [X]’.

We cannot however decide clearly where to insert the rule ‘昨日 \rightarrow yesterday’ as there is no matching non-terminal in the rule containing its parent in the input sentence (‘読んだ’). We use the term *floating* to describe words such as ‘yesterday’ in this example, i.e. for an input subtree matched to the source side of a rule, children of the input root that are not contained in the source side of the rule as terminals and cannot be inserted using fixed-position non-terminals in the rule.

Previous work deals with this problem by either using simple glue rules [17] or limiting rules in a way to avoid isolated floating children [88]. For example, it is possible to disallow the first rule in Figure 6.1 when translating a sentence such as that in Figure 6.2 with uncovered children (in this case the word ‘yesterday’). This method greatly reduces the expressiveness and flexibility of translation rules.

In our generalized model, we allow any number of terminals and non-terminals and permit arbitrarily many floating children in each rule. To our knowledge this is the first study to take this more comprehensive approach.

Note that in the case of constituency-based tree-to-tree translation it is possible to binarize the input tree and therefore gluing floating children becomes simpler, as we only have to choose between pre-insertion and post-insertion. In the depen-

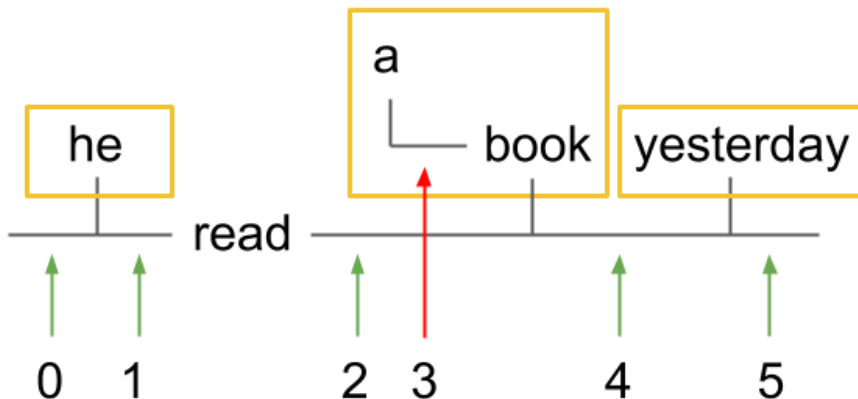


Figure 6.3: Possible insertion positions for flexible non-terminals with target-side head ‘read’. Allowed positions are shown in green and disallowed positions are shown in red. We do not allow insertion position 3 because it could allow a non-projective dependency structure.

dependency case it is in general much more difficult because we must order an arbitrarily large group of children sharing a common head.

6.3 Flexible Non-Terminals

In this study we propose flexible non-terminals in order to create generalized tree-to-tree translation rules that can overcome the problems described in the previous section. Rather than fixed insertion positions for child nodes, we instead consider multiple possible insertion positions and give features to each position. These are stored in a compact representation allowing for efficient decoding.

We define *flexible non-terminals* as non-terminals with multiple possible insertion positions and associated features. During decoding we select the most promising insertion position for each non-terminal.

6.3.1 Rule Augmentation

As is standard practice in phrase-based SMT, before translation we filter translation rules to those relevant to the input sentence. At this time, for each accepted rule we check the input sentence for floating children, and flexible non-terminals are added for each floating child.

We allow all insertion positions between the children (along with their descendants) of the target-side head for each floating child, including insertion before the first child and after the last child. We do not allow insertion positions between deeper descendants of the head to avoid non-projective dependencies. See Figure 6.3 for an example of allowed/disallowed positions.

Features are then set for each insertion position and these are used to determine the best insertion position during decoding (see Section 6.3.2). Figure 6.4 shows an example of the proposed rule augmentation.

6.3.2 Features

In previous work reordering is mostly decided by the combination of a standard distortion model and language model to score possible insertion positions. We instead consider the following four features and combine them during decoding to find the most appropriate insertion positions for floating children. All features are real numbers between 0 and 1.

Insertion Position Features

We first define a set of features to estimate the likelihood of each insertion position for some given non-terminal. The features for inserting the translation f of a source phrase into the target-side e of a rule at insertion position i are defined as follows, for surface forms (S) and POS tags (P):

- Reordering probability:
 $P_S(i | f, e), P_P(i | f, e)$
- Marginalized over target-side:
 $P_S(i | f), P_P(i | f)$

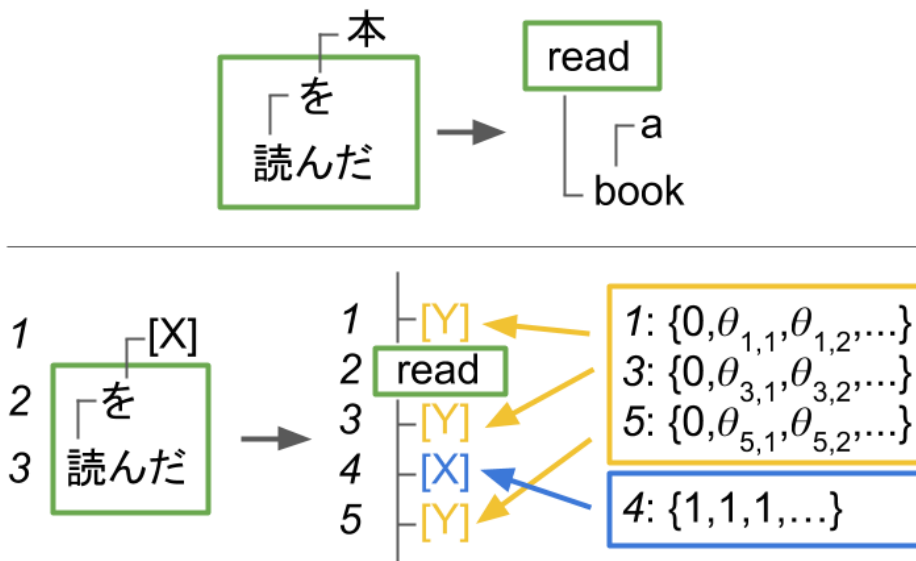


Figure 6.4: Example of translation rule with flexible non-terminals generated from the first parallel sentence in Figure 6.1. [X] has a fixed position (4) but [Y] can have multiple positions (1, 3, 5). Each position has an associated set of features shown in curly brackets, where $\theta_{i,j}$ is the j th feature for insertion position i . The first feature (0 or 1) shows whether the insertion position is unambiguous.

- Marginalized over source-side:

$$P_S(i | e), P_P(i | e)$$

The probabilities $P(i | X)$ are calculated by counting insertions of X in each position i across the whole training corpus (aligned and parsed bitext). The exact formula is given below, for position i (X is one of $\{f\}$, $\{e\}$ or $\{f, e\}$):

$$P(i | X) = \frac{\text{count}(i, X)}{\sum_j \text{count}(j, X)} \quad (6.1)$$

Instead of applying smoothing, in order to reduce sparsity issues we use both the full probability $P(i | f, e)$ and also probabilities marginalized over the source and target phrases. We also consider both probabilities trained on surface forms (S) and POS tags (P).

While traditional models use linear distance for i , this is impractical for long-distance reordering. Instead we restrict insertion types i to one of the following 6 types: first-pre-child, mid-pre-child, final-pre-child, first-post-child, mid-post-child, and final-post-child. These correspond to the first (first), last (final) or central (mid) children on the left (pre) or right (post) side of the parent word. We found this was more effective than using either linear distance or a binary (pre/post) position type.

Relative Position Feature

We also consider a relative position, or ‘swapping’ feature, inspired by the swap operation of classic lexical distortion [99].

Let T be the children of the root word of the target-side of a rule. We also include in T a pseudo-token M splitting the left and right children of the target-side root to differentiate between pre-insertion and post-insertion.

We first learn a model describing the probability of the translation of input phrase I appearing to the left ($P_L(I, t)$) or right ($P_R(I, t)$) of word t in the target-side of a translation rule. The probabilities are calculated by counting occurrences of I being translated to the left/right sides of t over the aligned and parsed training bitext.

The relative position feature is calculated by considering the relative position of the translation of I with all the target-side root children T . For each insertion position i , let $T_{i,L}$ be the $t \in T$ to the left of position i and $T_{i,R}$ the $t \in T$ to the right of position i . Then we have:

$$P(i | I, T) = \prod_{t \in T_{i,R}} P_L(I, t) \prod_{t \in T_{i,L}} P_R(I, t) \quad (6.2)$$

Left/Right Attachment Preference

We also set an attachment direction preference feature for each rule, specifying whether we prefer to insert the rule as a left child or right child of the root of a parent rule.

The attachment preference is determined by the position of the target-side of the rule in the target-side of the parallel sentence from which it was extracted. For example, in Figure 6.1 the rule ‘昨日 \rightarrow yesterday’ was extracted from a parallel sentence in which ‘yesterday’ was a right-side child of its head (‘saw’), so we set the attachment preference to ‘right’. In cases when we cannot determine the attachment preference (for example ‘read’ in the first rule in Figure 6.1), because it is the sentence root), we arbitrarily choose ‘right’.

Unambiguous Insertion Preference

In cases where we have a single unambiguous insertion position for a non-terminal (e.g. [X] in Figure 6.4), we set an additional binary feature to the value 1 (otherwise 0) to specify that this position is unambiguous. We found that a large positive weight is almost always given to this feature, which is to be expected as we would prefer to use fixed non-terminals if possible. We set all features related to insertion position choice to the maximum value (1).

6.3.3 Decoding

The flexible non-terminals that we are proposing can lead to some interesting challenges when it comes to decoding. A naive approach is to expand each translation rule containing flexible non-terminals into a set of ‘simple’ rules with fixed

non-terminals, and then apply classic decoding with cube-pruning.

However, this can be quite inefficient in practice. Due to the combinatorial aspect, a single rule can expand into a very large number of simple rules. It is common for our translation rules to have more than four flexible non-terminals, each with more than four possible insertion positions. Such rules will already generate hundreds of simple rules. In the most extreme cases, we may encounter rules having more than ten flexible non-terminals, leading to the generation of many millions of simple rules. This explosion of rules can lead to impractical decoding time and memory usage.

It is therefore important to make use of the compact encoding of many simple rules provided by the concept of flexible non-terminals in the decoding process itself. We use the decoding approach of right-hand lattices [25], an efficient way of encoding many simple rules. The idea is to encode the translation rules into a lattice form, then use this lattice to decode efficiently without the need to expand the flexible non-terminals explicitly.

Figure 6.5 shows how the concept of flexible non-terminals can be efficiently encoded into lattice form. The top half shows a target-side tree translation rule with flexible non-terminals X1, X2, X3 and X4 allowed to be inserted at any position that is a child of the word ‘a’, with the constraint that X1 comes before X2 and that X2 comes before X3. X5 is another flexible non-terminal that will be a child of the word ‘f’. The lower half shows a lattice compactly encoding all the possible combinations of non-terminal positions. Each path from the top-left to the bottom right in this lattice represents a choice for the insertion positions of the non-terminals. For example, the path marked with a dotted line represents the flattened sequence ‘b c X1 X2 a X3 X4 d e f X5 g’. The lattice form has only 48 edges, while an explicit enumeration of all combinations of insertion positions for the flexible non-terminals would force the decoder to consider ${}^8C_4 \times 3 \times 12 = 2520$ edges.

The insertion position features described above are added to the edges of the lattice. They are combined alongside the standard set of features, such as word penalty and language model score, using a standard log-linear model. The weights for the reordering features are tuned together with the standard features.

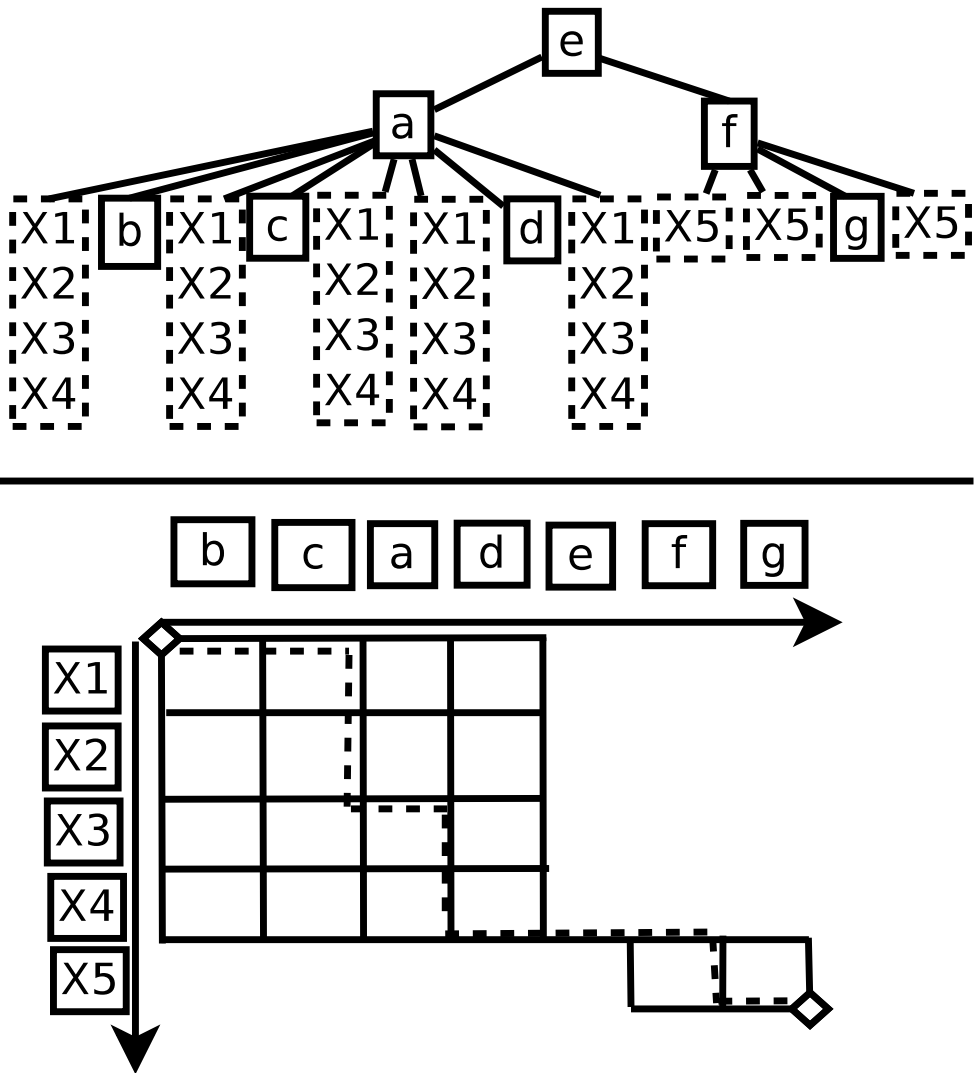


Figure 6.5: Example showing how a rule containing many flexible non-terminals is encoded into lattice form for decoding.

	JA-EN	EN-JA	JA-ZH	ZH-JA
Moses	18.09	27.48	27.96	34.65
Baseline	19.97	28.41	28.13	33.51
Flexible	21.23†	30.11†	29.42†	35.37†
+Pref	21.66‡	29.90†	29.48†	35.57‡
+Pref+Ins	21.47‡	30.03†	29.64†	35.71‡
+Pref+Ins+Rel	21.34†	29.99†	29.78‡	35.81‡

Table 6.1: Automatic evaluation of translation quality (BLEU). The results marked with † are significantly higher than the baseline system and those marked with ‡ are significantly higher than the proposed system with no insertion position features (‘Flexible’). Significance was calculated with bootstrapping for $p < 0.05$.

6.4 Experiments

6.4.1 Data and Settings

We performed translation experiments on four distant language pairs, Japanese-English (JA-EN), English-Japanese (EN-JA), Japanese-Chinese (JA-ZH) and Chinese-Japanese (ZH-JA), from the Asian Scientific Paper Excerpt Corpus (ASPEC).

Our experiments were conducted using KyotoEBMT, which is described in Chapter 2. The proposed non-terminal reordering features were added to the rules extracted with the baseline system. Dependency parsing was performed with the same tools as in Section 2.6 and alignment used the three-step pipeline described in Section 2.4.1.

6.4.2 Evaluation

As our baseline (‘Baseline’), we used the default settings and features of KyotoEBMT, allowing only fixed-position non-terminals. We dealt with floating children not covered by any other rules by adding glue rules similar to those in hierarchical SMT [17], joining floating children to the rightmost slots in the target-side parent. For reference, we also show results using Moses [49] with default settings

	JA-EN	EN-JA	JA-ZH	ZH-JA
Moses	63.97	68.37	79.03	77.25
Baseline	65.10	74.78	78.00	77.86
Flexible	69.94†	77.11†	80.44†	81.33†
+Pref	70.73‡	76.85†	80.43†	81.79‡
+Pref+Ins	70.85‡	77.01†	80.65†	82.05‡
+Pref+Ins+Rel	70.69‡	76.93†	80.51†	81.95‡

Table 6.2: Automatic evaluation of translation quality (RIBES). The results marked with † are significantly higher than the baseline system and those marked with ‡ are significantly higher than the proposed system with no insertion position features (‘Flexible’). Significance was calculated with bootstrapping for $p < 0.05$.

and distortion limit set to 20 (‘Moses’).

The proposed system (‘Flexible’) adds flexible non-terminals with multiple insertion positions, however we do not yet add the insertion choice features. This means that the insertion positions are in practice chosen by the language model. Note that we do not get a substantial hit in performance by adding the flexible non-terminals because of their compact lattice representation. The systems ‘+Pref’, ‘+Pref+Ins’ and ‘+Pref+Ins+Rel’ show the results of adding insertion choice position features (left/right preference, insertion position choice, relative position choice).

We give translation scores measured in BLEU [74] and RIBES [44], which is designed to reflect quality of translation word order more effectively than BLEU. The translation evaluation is shown in Table 6.1 and Table 6.2.

6.5 Discussion and Error Analysis

The experimental results showed a significantly positive improvement in terms of both BLEU and RIBES over the baseline tree-to-tree system. The baseline system uses fixed non-terminals and is competitive with the most popular string-to-string system (Moses).

	JA-EN	EN-JA	JA-ZH	ZH-JA
% rules with flexible NTs	53.2	70.4	55.7	61.2
Average flexible NTs per rule	0.973	1.11	0.977	1.05
% all NTs that are flexible	48.0	48.6	54.5	56.1
% selected NTs that are flexible	32.2	35.1	40.5	58.4

Table 6.3: Results of non-terminal (NT) matching analysis.

The extensions of the proposed model (adding a variety of features) also all showed significant improvement over the baseline, and approximately half of the extended settings performed significantly better than the core proposed model. It is unclear however which of the extended settings is the most effective for all language pairs. There are a number of factors such as parse quality, corpus size and out-of-vocabulary occurrence that could affect the potential value of these features. Furthermore, Japanese is strongly left-branching (head-final), so the left/right preference distinction is likely to be less useful than for English and Chinese, which contain both left-branching and right-branching structures.

Compared to the baseline, the flexible non-terminals gave around a 1.2–1.9 BLEU improvement at the cost of only a 30% increase in decoding time (approximately 2.04 vs. 2.66 seconds per sentence). This is made possible by the compact non-terminal representation combined with lattice decoding.

6.5.1 Non-Terminal Matching Analysis

We found that roughly half of all our translation rules were augmented with flexible non-terminals, with one flexible non-terminal added per rule on average. This led to roughly half of non-terminals having flexible insertion positions. The decoder chose to use ambiguous insertion positions between 30%–60% of the time (depending on language pair), allowing for many more new translation hypotheses than the baseline system. For detailed results, see Table 6.3.

6.5.2 Translation Examples

The following translation is an example of an improvement achieved by using the proposed flexible non-terminals. There were multiple word order errors in the baseline translation that impeded understanding, and these have all been corrected.

- **Input:** 磁場入口と出口の温度差により生ずる磁性流体の圧力差と流速を測定した。
- **Reference:** The pressure difference and the flow velocity of the magnetized fluid caused by the temperature difference between the inlet and outlet of the magnetic field were measured.
- **Baseline:** We have measured the pressure difference and flow rate of a magnetic fluid generated by an entrance of a magnet and an exit temperature, and the difference between.
- **Proposed:** The pressure difference and the flow rate of a magnetic fluid generated by the temperature difference between the magnetic field inlet and exit were measured.

There are also cases where the proposed model decreases translation quality. In the example below, the proposed system output was selected by the decoder since it had a higher language model score than the baseline output, despite having incorrect word order. The incorrect translation was made available by the increased flexibility of the proposed model, and selected because the LM feature had a higher impact than the insertion position features.

- **Input:** このソフトウェアのR 5バージョンの特徴, 利用マニュアルと設計文書をまとめた。
- **Reference:** The characteristics of R5 version of this software, instruction manual, and design document were summarized.
- **Baseline:** The R5 version of this software features, the manual for the utilization and design documents are summarized.
- **Proposed:** This software design documents of R5 version features, the manual for the utilization and summarized.

6.6 Summary

In this study, we have proposed flexible non-terminals for dependency tree-to-tree translation. Flexible non-terminals allow multiple insertion positions to be expressed compactly and selected with features based on both source and target syntax. We have shown that a significant improvement in BLEU and RIBES scores can be gained by using the proposed model to increase the generality of dependency tree-to-tree translation rules.

The work in this chapter has improved reordering, perhaps the most important problem in translation between distant language pairs. It is clear that target-side dependency syntax is valuable for choosing word order, as we are able to select target-side non-terminal insertion positions in a way that ensures the syntactic fluency of translations.

Chapter 7

Neural Networks: The Future?

7.1 Introduction

As we reach the end of our study on target-side syntax, we consider possible directions for future research. The approaches described in this thesis are based on Statistical Machine Translation, which until the last couple of years has been the major focus of MT research. Recently approaches to translation based on neural networks, known as Neural Machine Translation (NMT), have become popular and have already demonstrated impressive results. In this chapter we ask how our work fits into the world of neural translation.

First we describe the two current major approaches of incorporating neural networks into MT systems: language models and translation models.

7.1.1 Language Models

One of the first applications of neural networks and deep learning to NLP was the Recurrent Neural Network Language Model (RNNLM) [62]. RNNLMs are based on recurrent neural networks, which have a cyclic structure allowing the recursive combination of states.

The RNNLM has a generative story similar to a standard n -gram language model, i.e. the next word in a sentence is that with the maximum likelihood given the previously seen context. For classic n -gram LMs, the previous $n - 1$ words are used as context in order to reduce model sparsity as the probabilities are

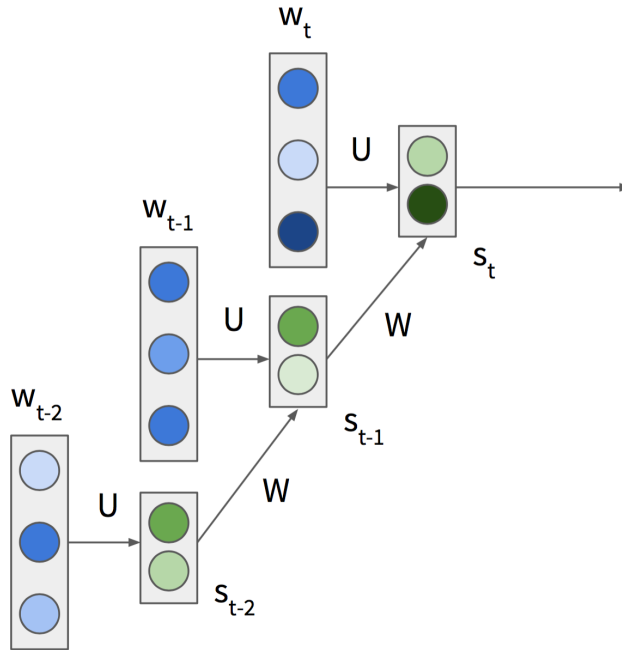


Figure 7.1: Structure of a RNNLM. The sequence of word embeddings w_0, \dots, w_{t-1}, w_t are combined recursively to form a state s_t at time t . The same matrices W and U are used for each combination.

estimated from limited data. For RNNLMs however, the next word is conditioned on a ‘context state’, which is updated recursively based on the previous context state and the next word.

Figure 7.1 shows the unrolled structure of a RNNLM. The context state at time t , denoted by s_t , is created by combining the previous state s_{t-1} and the current word vector w_t with matrices W and U respectively. The word vectors are usually either one-hot vectors or word embeddings [61]. The current state s_t is used as the input to a network generating the next word. In practice this is often simply a softmax layer over the vocabulary.

An additional factor that has led to the success of recurrent models is the Long Short-Term Memory (LSTM) unit, a more powerful NN unit than the neuron with a structure designed to model ‘memory’ when combining recurrent states. This

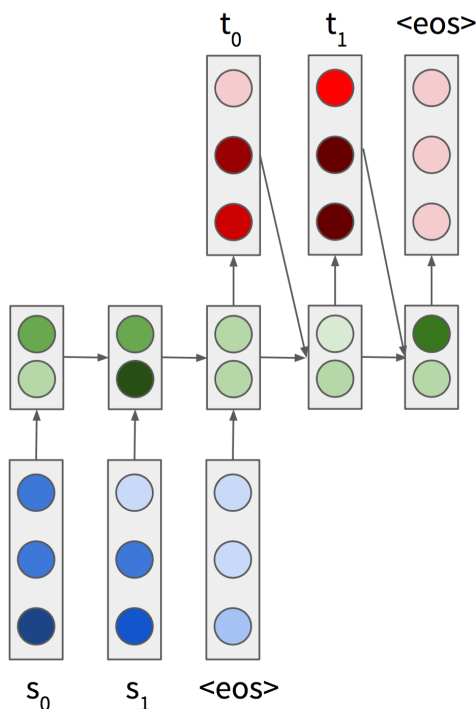


Figure 7.2: Example of a sequence-to-sequence translation model. Target words t_0, t_1 (red) are generated from internal states (green), which in turn are the result of combining source words s_0, s_1 (blue) with already translated target words.

allows LSTMs to improve the recurrent nature of RNNs as they can successfully combine a long sequence of inputs, effectively ‘remembering’ or ‘forgetting’ each previous input. An alternative to the LSTM unit is the simpler Gated Recurrent Unit (GRU) unit, which has been shown to have a similar ability to improve recurrent networks [22].

7.1.2 Translation Models

The simplest approach to building neural translation models has been in fact the most successful. Sequence-to-sequence methods [97] model translation as a black box that learns to output a sequence of tokens given an input sequence.

The simplest sequence-to-sequence model is shown in Figure 7.2. The source

words s are combined recurrently to form a context state until the end-of-sentence ‘<eos>’ token is seen. The system then outputs target tokens t based on the current context state until the end-of-sentence token is outputted. The context state is updated recurrently based on the previous context state and the last outputted target word.

The idea is similar to Hidden Markov Models (HMMs), which produce output symbols based on a fixed history of input symbols, however the major difference is that RNNs are able to consider arbitrarily long history without the model sparsity penalty faced by HMMs. Sequence-to-sequence models can be improved by adding an ‘attention mechanism’ [3], which builds the context state from a weighted combination of the source words in order to emulate the notion of SMT word alignments and reordering.

7.1.3 Hybrid Syntax with Recursive NNs

Recurrent neural networks are similar to n -gram language models in that they consider a left-to-right sequence of words. Recursive neural networks [94] are the syntactic equivalent to recurrent networks in that they instead build context based on a tree structure.

An example of a recursive neural network is shown in Figure 7.3. The sentence is first parsed¹ and word vectors are combined according to this parse tree structure to build a sentence state.

Recursive neural networks have been shown to produce high-quality word embeddings [57] that are syntactically motivated. It is also possible to create end-to-end translation systems using recursive neural networks [53]. While this approach to hybridizing syntactic approaches and neural networks is interesting, such methods have not yet been shown to outperform recurrent NNs. This could be due to the reliance on an input parse, which cannot be guaranteed to be of high quality.

¹This example uses a binarized constituency parse but dependency parses work similarly.

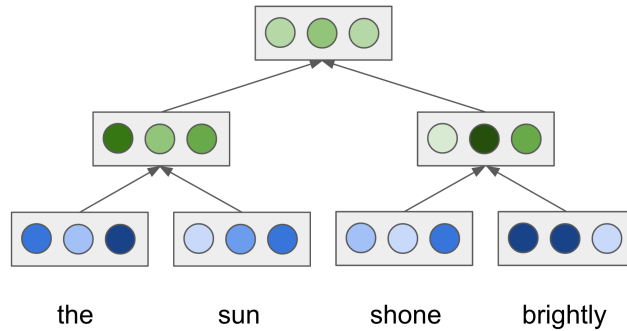


Figure 7.3: Building a syntactically motivated sentence state with a recursive neural network. Words are combined hierarchically with the same structure as a precomputed syntax tree. In this example we use the binarized constituency parse: (S (NP (DT the) (NN sun)) (VP (VBP shone) (ADVP (RB brightly))))).

7.1.4 Overview

The core section of this thesis concentrated on two major approaches to improving dependency tree-to-tree MT: post-editing with language models and syntax-based reordering. We now consider two simple yet effective neural network extensions to these approaches.

In Section 7.2 we employ state-of-the-art neural language models to improve post-editing (specifically reranking). Section 7.3 considers neural network features for selecting flexible non-terminal insertion positions.

7.2 Neural Networks for Reranking

In our language modeling experiments we concentrated on post-editing. One of the major benefits of post-editing methods is that they are decoder independent and therefore compatible with both SMT and NMT systems. Indeed our approach in Chapter 5 was shown to be effective with a baseline that contained neural translation components.

Another possible decoder-independent application of neural networks is to use them to rerank k -best translation output. We performed a simple experiment to

assess whether neural network features are able to improve our baseline dependency tree-to-tree SMT system, which uses features mainly inspired by syntax.

One of the major benefits of a reranking step is that it allows us to use features that are more expensive to calculate. For example, a RNNLM model score takes considerably longer to calculate than for a standard language model, thus making such features difficult to consider during decoding. More importantly, we make heavy use of language model state optimizations during decoding and these are not compatible with a neural language model. Decoding with neural LMs is beyond the scope of this thesis.

7.2.1 Reranking Framework

We introduced a reranking step into the KyotoEBMT pipeline, taking the k -best system output and selecting the optimal translation based on a log-linear score similar to that used during decoding. For each translation in the k -best list, we calculated a number of additional features (described in Section 7.2.2) and appended these to the original features in the k -best list.

We then reran the tuning pipeline on the k -best list augmented with additional features. The tuning algorithm and settings were the same as for standard tuning (see Section 2.5.3 for details). This retuning step was added because some original features, such as the sentence length, could interact with the additional features. It is therefore necessary to retune in order to find the optimal combination of the extended feature set.

7.2.2 Features

In our experiments we added the following additional neural network features:

- Monolingual RNNLM (word-based)
- Bilingual RNNLM (word-based)
- Monolingual RNNLM (character-based, Japanese/Chinese only)
- Bilingual RNNLM (character-based, Japanese/Chinese only)

We added character-based models as additional features for Japanese and Chinese, as these languages do not have marked word boundaries. There are two major advantages of character-based models: we do not rely on word segmentation tools, which are not always accurate; and models can be more compact as we have a smaller vocabulary size, i.e. the number of unique characters, not words.

7.2.3 Experiments

The monolingual RNNLM model was trained with the RNNLM toolkit² using a hidden layer size of 200. We held out 5000 sentences from the training data for validation.

The bilingual RNNLM was the sequence-to-sequence translation model with attention [3]. We used the open source implementation in the GroundHog/Theano framework³. We used a hidden layer size of 1000 and training was performed with a minibatch size of 64 and the ADADELTA algorithm [107] ($\rho = 0.95$, $\epsilon = 10^{-6}$). The vocabulary size was reduced to 20K for the word-segmented model. The models took two to four days each to train on a GPU.

For comparison, our experiments used exactly the same settings as those described in Section 2.6 for the WAT14/WAT15 experiments. We used the 1000-best translations for reranking and the results are shown in Table 7.1.

7.2.4 Conclusion

The results show that a clear improvement was achieved by using the additional neural network language model features.

It is interesting to note that, despite the fact the bilingual neural models on their own are not as powerful as our system for end-to-end translation, they appear to be very useful for reranking our system output. This suggests that the neural models learn different types of information to our system (and make different errors), since they represent a completely distinct approach to translation. In the future it would be interesting to try to understand concretely the nature of these differences.

²<http://rnnlm.org>

³<https://github.com/lisa-groundhog/GroundHog>

Language Pair	System	BLEU	RIBES
JA-EN	Base-Phrase	18.45	64.51
	Base-Hiero	18.72	65.11
	WAT14	20.60	70.12
	WAT14+Rerank	21.07	69.90
	WAT15	21.31	70.65
	WAT15+Rerank	22.89	72.46
EN-JA	Base-Phrase	27.48	68.37
	Base-Hiero	30.19	73.47
	WAT14	29.76	75.21
	WAT14+Rerank	31.09	75.96
	WAT15	30.69	76.78
	WAT15+Rerank	33.06	78.95
JA-ZH	Base-Phrase	27.96	78.90
	Base-Hiero	27.71	80.91
	WAT14	27.21	79.13
	WAT14+Rerank	27.67	78.83
	WAT15	29.99	80.71
	WAT15+Rerank	31.40	82.70
ZH-JA	Base-Phrase	34.65	77.25
	Base-Hiero	35.43	81.04
	WAT14	33.57	80.10
	WAT14+Rerank	34.75	80.26
	WAT15	36.30	81.97
	WAT15+Rerank	38.53	84.07

Table 7.1: Evaluation results (BLEU/RIBES) for our WAT14/WAT15 submissions after reranking with neural network features.

Our simple experiment showed that neural network language models are able to complement the information learned by our system and therefore it seems also worth considering their use during decoding. The ideal approach would be to design a tree-based neural LM in order to combine the modeling power of syntactic and neural methods.

7.3 Improving Flexible Non-Terminals with NNs

Chapter 6 described flexible non-terminals and their effectiveness in ordering target-side dependency trees. We used traditional lexicalized language model features to guide the choice of insertion position during decoding.

In this section we replace these SMT-style features with a neural network that selects insertion positions before decoding. The proposed approach improves upon the translation quality achieved in Chapter 6. An additional benefit of this method is that the decoder search space is greatly reduced and we show that this can lead to a considerable reduction in decoding time.

7.3.1 Insertion Position Selection

Our tree-to-tree translation system makes use of translation hypotheses encoding multiple insertion positions for each non-terminal on the target-side. For each source-side word I to be inserted, we extract both source-side and target-side features for each possible target-side insertion point J . These features are used as the input to a neural network that predicts the optimal non-terminal insertion position.

We use the parent of I (P_s) and the linear distance between I and P_s (D_s) as source-side features. Target-side features are the previous (S_p) and next (S_n) siblings of J , the parent of the insertion position (P_t) and the number of siblings between J and P_t (D_t). We express the number of intermediate siblings as negative when J is to the left of P_t .

The word-based features are converted into 220-dimensional vector representations by concatenating embeddings for the surface form (dimension 200), part-of-speech (dimension 10) and dependency type (dimension 10). These are combined

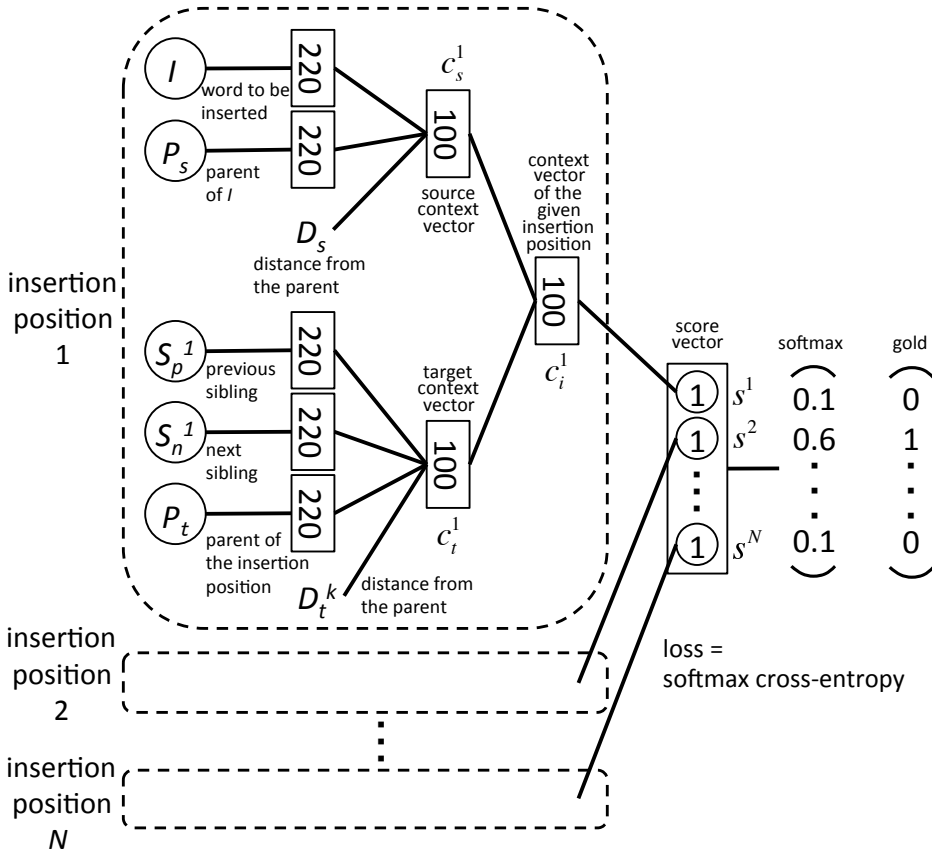


Figure 7.4: Neural network for insertion position selection. The numbers inside boxes show the dimensions of the corresponding vectors.

to form source and target vectors, which are again combined into a single context vector for each position. We use a *tanh* layer for each combination, compacting the representations to 100 dimensions. All the links between layers are fully-connected, however we use dropout (50%) to avoid overfitting.

Figure 7.4 shows the design of the neural network combining these features to decide the insertion position. At inference time, the network is applied to all insertion positions to obtain their scores, which are normalized by a softmax function. The position with the highest corresponding score is selected.

7.3.2 Experiments

We first conducted an experiment to assess the insertion position selection accuracy of the proposed system. Training data for the neural network model was extracted from a word-aligned parallel corpus (ASPEC). We removed each aligned target-side node from its tree and used the original position as gold data. In total we extracted approximately 16M/6M examples for JA \leftrightarrow EN/JA \leftrightarrow ZH respectively, 160K/60K of which were used for each of development and test sets. Training was conducted for 100 epochs.

The test set insertion position accuracies measured were: 97.08 (JA–EN), 97.72 (EN–JA), 96.51 (JA–ZH) and 97.99 (ZH–JA). The observed accuracies suggested that the proposed method is promising for selecting flexible non-terminal insertion positions. The results were highest for Japanese as the target language, since most children are inserted to the left of their parents (strongly head-final).

We then conducted an experiment to measure the resulting translation quality (BLEU/RIBES) and decoding time when using only the insertion positions predicted by the proposed model. We used the same data and baseline system settings as in Section 6.4. The systems we compared were as follows:

- No Flexible: Simple glue rules (‘Baseline’ in Section 6.4)
- Baseline: Flexible non-terminals (‘+Pref’ in Section 6.4)
- Proposed: Single best insertion position caused by neural network model

Table 7.2 gives the results of the translation experiment. The proposed method achieved significantly higher automatic evaluation scores than the baseline for all language pairs tested, except BLEU for EN–JA. Furthermore, the decoding time was reduced by about 60% relative to the baseline, since we greatly reduced the translation search space by not considering insertion positions not predicted by the NN model.

7.3.3 Conclusion

Our experiments in this section have shown that neural networks can be integrated successfully into our dependency tree-to-tree reordering model. Furthermore, it

	JA-EN			EN-JA		
	BLEU	RIBES	Time	BLEU	RIBES	Time
No Flexible	20.28	65.08	1.00	28.77	75.21	1.00
Baseline	21.61	69.82	6.28	30.57	76.13	3.30
Proposed	22.07	70.49	2.25	30.50	76.69	1.27

	JA-ZH			ZH-JA		
	BLEU	RIBES	Time	BLEU	RIBES	Time
No Flexible	24.85	66.60	1.00	30.51	73.08	1.00
Baseline	28.79	78.11	5.16	34.32	77.82	5.28
Proposed	29.83	79.73	2.21	34.71	79.25	1.89

Table 7.2: Translation results using neural networks to determine non-terminal insertion positions. Bold type signifies results significantly better than the baseline ($p < 0.01$).

seems that they are even more effective than our original SMT-style features at predicting non-terminal insertion positions. We also saw a significant increase in decoding speed facilitated by the reduction of search space size.

In this first exploration we restricted context to individual words (subtree heads), however it could be more promising to generalize this to complete subtrees. For example, the information in the context subtree ‘in fact’ is more informative than simply ‘in’. This could be achieved by employing recursive neural networks to encode context. It would also be interesting to combine the decoding time approach in Chapter 6 to use the NN prediction scores to guide decoding, although this would negate the speed increase made possible by the proposed approach.

7.4 Summary

Neural MT is fundamentally different from SMT. In this chapter we have shown that this fact opens new avenues for extending our models with neural components.

We have proposed a number of post-editing and reranking approaches (see

Chapters 3–5) that are decoder independent, allowing them to be applied trivially to neural decoders. We can also expand these reranking frameworks by integrating neural network components, such as neural LMs, as demonstrated in Section 7.2. We have shown that adding neural features to reranking was able to improve significantly improve the quality of KyotoEBMT, our tree-to-tree SMT system.

Neural features can also be used to improve the quality of our proposed syntactic reordering model. We have shown in Section 7.3 that selecting insertion positions of flexible non-terminals can be achieved more efficiently and accurately with neural networks than the traditional lexicalized LM features proposed in Chapter 6. This is a clear example of the possibility of building hybrid systems.

While the construction of purely neural syntactic approaches is beyond the scope of this thesis, our survey of previous studies on recursive neural networks has shown that there is clear potential in such approaches. The important question remaining is whether syntactically motivated neural methods can be more effective than end-to-end neural systems that are not explicitly fed syntactic information.

Chapter 8

Conclusion

8.1 Overview

The goal of this thesis has been to explore the effectiveness of target-side dependency syntax in the improvement of statistical machine translation. We have asked how to design tree-to-tree models robustly and efficiently, considering their complexity and reliance on parsing quality, and asked in which areas of the translation pipeline target-side syntax can be exploited most effectively.

To answer these questions, we have performed empirical studies comparing the effectiveness of traditional phrase-based systems and approaches exploiting the target-side syntax of a tree-to-tree system. As our tree-to-tree experimental framework we have developed KyotoEBMT, a state-of-the-art dependency forest-to-tree translation system, which is described in detail in Chapter 2.

The approaches that we have considered in this thesis can be classified into two major categories: decoding-time approaches with tree reordering rules, and post-editing approaches with tree-based language models.

We began our exploration of post-editing approaches with a simple dependency-based language model for editing function words in Chapter 3. The observation that the intended meaning of a translation is preserved more closely in structured than flat output allowed us to correct function word errors in tree-to-tree translations. While we showed that this method was able to give a significant improvement in translation quality as judged by human raters, the scope of this approach

was not sufficient.

In Chapter 4, we described the design and implementation of a generalized dependency tree language model. Using this model in post-editing we were able to improve the long-range dependencies in even flat string output by parsing translations. It was clear however that parsing quality and the lack of source-side context were serious drawbacks to the proposed approach. We therefore attempted in our work described in Chapter 5 to overcome these issues by training a language model on projected source-side dependency syntax as opposed to training directly on the target side. We found that this approach was considerably more effective, giving strongly positive results for all twenty language pairs tested.

We considered the enhancement of translation and reordering rules as our second major application of target-side syntax. In Chapter 6 we proposed non-terminals with variable insertion positions that enable the design of flexible dependency tree-to-tree translation rules. Our experiments showed that improving the expressiveness of the reordering model led to a considerable increase in translation quality.

It is difficult to say which of these approaches is more effective as they focus on tackling difficulties for translation caused by distinct linguistic phenomena: morphology and word ordering. We can expect the greatest improvement to be seen for language pairs where both morphology and word order differ greatly between source and target languages.

A fundamental difference between our two proposed approaches is the use of post-editing (language modeling) versus decoder integration (reordering). This means the two approaches are independent and can be combined trivially to gain potentially further improvement. We believe however that it makes sense to tune on separate evaluation metrics (human evaluation or BLEU respectively) depending on whether the optimization is conducted manually or automatically. While this makes it less clear which of the two approaches is the most effective, by performing independent tuning we are able to achieve the the optimal performance for each model individually. It should also be noted that we used a larger corpus to train the monolingual language models than the bilingual systems owing to the availability of data.

8.2 Future Work

8.2.1 Traditional Approaches

It is clear that target-side syntax is of value to statistical machine translation systems despite its drawbacks of complexity and reliance upon upstream linguistic analysis. We were able to achieve state-of-the-art results by applying our methods to traditional translation tasks for a variety of language pairs.

In the coming years we can expect to see further improvement in monolingual analysis (particularly parsing) of major languages and this would further increase the value of syntax-based approaches. However there will remain low-resource languages for which we do not reach an adequate level of parsing quality for the foreseeable future. There is therefore value in pursuing work that adds robustness to syntax-based approaches, such as forest-based methods and models that exploit rich source language syntax similar to our work in Chapter 5.

The majority of syntax-based MT research is motivated by monolingual linguistic theory, which can be challenging at times to adapt to the bilingual setting of translation. Joint alignment and parsing models, syntax projection, and unsupervised learning of bilingual grammars could allow for the construction of more robust translation rules. This area has seen very little exploration and could be an exciting direction for future research.

8.2.2 Neural Networks

While at the time of writing statistical methods are still leading the field for distant language pairs such as Japanese–English, neural approaches have recently overtaken SMT for a number of language pairs, and their development continues to grow at a rapid pace. It is therefore natural that future work must consider whether it is possible to integrate syntax-based approaches into neural translation frameworks.

In Chapter 7 we begin this exploration by suggesting a variety of hybrid approaches. It can be relatively simple to integrate neural features into traditional SMT systems and we have shown in our preliminary experiments that these can give clear quality improvements. While previous studies have shown that pure neu-

ral approaches tend in general to outperform hybrid methods, there are cases such as low-resource scenarios where hybrid approaches are likely to remain effective.

It remains to be seen to what extent pure neural systems can make use of syntax. Long Short-Term Memory (LSTM) cells combined to form ‘attention’ mechanisms can be considered a parallel of SMT alignment and reordering, but can these benefit from being fed with existing syntactic structures?

8.2.3 Final Words

The core hypothesis of tree-based MT is that syntax is a valuable abstraction for modeling translation. We have shown in this thesis that syntax (in particular dependency grammar) is certainly valuable, however it remains to be seen whether our current linguistic theories can express the optimal abstractions for translation.

Future research will further generalize and abstract both syntactic and semantic representations of natural languages. Indeed, recent advancements in neural network research have hinted that linguistic theories are already being learned and not taught. Once we are able to understand abstractions learned using unsupervised methods, we can aim improve existing linguistic theories to increase their impact on MT applications.

Bibliography

- [1] D. Arnold and L. Sadler. EuroTra: An Assessment of the Current State of the ECs MT Programme. In *Working Papers in Language Processing*, 1991.
- [2] E. Avramidis and P. Koehn. Enriching Morphologically Poor Languages for Statistical Machine Translation. In *Proceedings of ACL-08: HLT*, pages 763–770, Columbus, OH, USA, June 2008. Association for Computational Linguistics.
- [3] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014.
- [4] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, March 2003.
- [5] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71, March 1996.
- [6] O. Bojar, R. Chatterjee, C. Federmann, B. Haddow, M. Huck, C. Hokamp, P. Koehn, V. Logacheva, C. Monz, M. Negri, M. Post, C. Scarton, L. Specia, and M. Turchi. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

- [7] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large Language Models in Machine Translation. In *EMNLP*, pages 858–867, 2007.
- [8] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, June 1990.
- [9] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, June 1993.
- [10] C. Callison-Burch. Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon’s Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP ’09*, pages 286–295, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
- [11] C. Callison-Burch, C. Bannard, and J. Schroeder. Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL ’05*, pages 255–262, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.
- [12] V. Chahuneau, E. Schlinger, N. A. Smith, and C. Dyer. Translating into Morphologically Rich Languages with Synthetic Phrases. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1677–1687, Seattle, WA, USA, October 2013. Association for Computational Linguistics.
- [13] E. Charniak and M. Johnson. Coarse-to-fine N-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL ’05*, pages 173–180. Association for Computational Linguistics, 2005.

- [14] C. Chelba. A Structured Language Model. In *Proceedings of the 8th Conference on European Chapter of the Association for Computational Linguistics*, pages 498–500. Association for Computational Linguistics, 1997.
- [15] S. F. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics, ACL '96*, pages 310–318, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.
- [16] C. Cherry and G. Foster. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT '12*, pages 427–436, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.
- [17] D. Chiang. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, 2005.
- [18] D. Chiang. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2):201–228, June 2007.
- [19] D. Chiang. Learning to Translate with Source and Target Syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden, July 2010. Association for Computational Linguistics.
- [20] D. Chiang. Hope and Fear for Discriminative Training of Statistical Translation Models. *Journal of Machine Learning Research*, 13(1):1159–1187, April 2012.
- [21] N. Chomsky. *Syntactic Structures*. Mouton, The Hague, Netherlands, 1957.
- [22] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555, 2014.

- [23] B. Cowan and M. Collins. A Discriminative Model for Tree-to-Tree Translation. In *EMNLP*, pages 232–241, 2006.
- [24] F. Cromieres and S. Kurohashi. Efficient Retrieval of Tree Translation Examples for Syntax-Based Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11*, pages 508–518, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.
- [25] F. Cromières and S. Kurohashi. Translation Rules with Right-Hand Side Lattices. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 577–588, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [26] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.
- [27] C. Ding and M. Yamamoto. A Generative Dependency N-gram Language Model: Unsupervised Parameter Estimation and Application. *Journal of Natural Language Processing*, 21(5):981–1009, 2014.
- [28] J. Elming. Transformation-Based Corrections of Rule-Based MT. In *EAMT 11th Annual Conference*, 2006.
- [29] M. Faruqui, Y. Tsvetkov, G. Neubig, and C. Dyer. Morphological Inflection Generation Using Character Sequence to Sequence Learning. 2016.
- [30] L. Fleiss, B. Levin, and M. C. Paik. The Measurement of Interrater Agreement. In *Statistical Methods for Rates and Proportions (2nd ed)*, pages 212–236. Wiley, 1981.
- [31] M. Galley, M. Hopkins, K. Knight, and D. Marcu. What’s in a translation rule? In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2004.

- [32] E. Garmash and C. Monz. Dependency-Based Bilingual Language Models for Reordering in Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1689–1700, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [33] I. Goto, M. Utiyama, E. Sumita, and S. Kurohashi. Preordering using a Target-Language Parser via Cross-Language Syntactic Projection for Statistical Machine Translation. *ACM Transactions on Asian & Low-Resource Language Information Processing*, 14(3):13, 2015.
- [34] J. Graehl and K. Knight. Training Tree Transducers. In *HLT-NAACL 2004*, pages 105–112, Boston, MA, USA, 2004. Association for Computational Linguistics.
- [35] J. Gubbins and A. Vlachos. Dependency Language Models for Sentence Completion. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1405–1410, 2013.
- [36] E. Hasler, B. Haddow, and P. Koehn. Margin Infused Relaxed Algorithm for Moses. *Prague Bulletin of Mathematical Linguistics*, 96:69–78, 2011.
- [37] K. Heafield. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, 2011.
- [38] K. Heafield, H. Hoang, P. Koehn, T. Kiso, and M. Federico. Left Language Model State for Syntactic Machine Translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 183–190, San Francisco, CA, USA, December 2011.
- [39] K. Heafield, P. Koehn, and A. Lavie. Language Model Rest Costs and Space-Efficient Storage. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1169–1178, Jeju Island, Korea, July 2012.
- [40] M. Hopkins and J. May. Tuning As Ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP ’11,

- pages 1352–1362, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.
- [41] A. Huang, T. Kuo, Y. Lai, and S. Lin. Discovering Correction Rules for Auto Editing. *International Journal of Computational Linguistics and Chinese Language Processing*, 15(3-4), 2010.
- [42] L. Huang, K. Knight, and A. Joshi. A Syntax-Directed Translator with Extended Domain of Locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 1–8, New York City, NY, USA, June 2006. Association for Computational Linguistics.
- [43] R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. Bootstrapping Parsers via Syntactic Projection Across Parallel Texts. *Natural Language Engineering*, 11(3):311–325, September 2005.
- [44] H. Isozaki, T. Hirao, K. Duh, K. Sudoh, and H. Tsukada. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952, Cambridge, MA, USA, October 2010. Association for Computational Linguistics.
- [45] W. Jiang, Y. Lv, Y. Liu, and Q. Liu. Effective Constituent Projection across Languages. In *Proceedings of the 23rd International Conference on Computational Linguistics*.
- [46] D. Kawahara and S. Kurohashi. A Fully-lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, HLT-NAACL '06*, pages 176–183, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.
- [47] D. Kawahara and S. Kurohashi. A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis. In *Proceedings of the Main*

- Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 176–183. Association for Computational Linguistics, 2006.
- [48] P. Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the 10th Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT.
- [49] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.
- [50] P. Koehn, F. Och, and D. Marcu. Statistical Phrase-Based Translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics, Association for Computational Linguistics.
- [51] U. Lerner and S. Petrov. Source-Side Classifier Preordering for Machine Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, 2013.
- [52] Z. Li and S. Khudanpur. A Scalable Decoder for Parsing-Based Machine Translation with Equivalent Language Model State Maintenance. In *Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation (SSST-2)*, pages 10–18, Columbus, OH, USA, June 2008. Association for Computational Linguistics.
- [53] S. Liu, N. Yang, M. Li, and M. Zhou. A Recursive Recurrent Neural Network for Statistical Machine Translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1491–1500, Baltimore, MD, USA, June 2014. Association for Computational Linguistics.

- [54] Y. Liu, Q. Liu, and S. Lin. Tree-to-String Alignment Template for Statistical Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia, July 2006. Association for Computational Linguistics.
- [55] W. N. Locke and A. D. Booth. The Georgetown-IBM Experiment. In *Machine Translation of Languages*, pages 124–135. MIT Press, Cambridge, MA, USA, 1955.
- [56] A. Lopez. Hierarchical Phrase-Based Translation with Suffix Arrays. In *Proceedings of EMNLP-CoNLL*, pages 976–985, 2007.
- [57] T. Luong, R. Socher, and C. D. Manning. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, pages 104–113, 2013.
- [58] J. B. Mariño, R. E. Banchs, J. M. Crego, A. de Gispert, P. Lambert, J. A. R. Fonollosa, and M. R. Costa-Jussà. N-Gram-Based Machine Translation. *Computational Linguistics*, 32(4):527–549, December 2006.
- [59] A. Menezes and C. Quirk. Syntactic Models for Structural Word Insertion and Deletion during Translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 735–744, Honolulu, Hawaii, USA, October 2008. Association for Computational Linguistics.
- [60] H. Mi, L. Huang, and Q. Liu. Forest-Based Translation. In *Proceedings of ACL-08: HLT*, pages 192–199, Columbus, OH, USA, June 2008. Association for Computational Linguistics.
- [61] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013.
- [62] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent Neural Network Based Language Model. In *INTERSPEECH 2010, 11th*

Annual Conference of the International Speech Communication Association, pages 1045–1048, 2010.

- [63] S. Mori, M. Nishimura, and N. Itoh. Improvement of a Structured Language Model: Arbori-Context Tree. In *EUROSPEECH 2001 Scandinavia, 7th European Conference on Speech Communication and Technology*, pages 713–716, 2001.
- [64] M. Nagao. A Framework of a Mechanical Translation Between Japanese and English by Analogy Principle. In *Proceedings of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, New York, NY, USA, 1984. Elsevier North-Holland, Inc.
- [65] T. Nakazawa and S. Kurohashi. Alignment by Bilingual Generation and Monolingual Derivation. In *Proceedings of COLING 2012*, pages 1963–1978, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.
- [66] T. Nakazawa, H. Mino, I. Goto, S. Kurohashi, and E. Sumita. Overview of the 1st Workshop on Asian Translation. In *Proceedings of the 1st Workshop on Asian Translation (WAT2014)*, pages 1–19, Tokyo, Japan, October 2014.
- [67] T. Nakazawa, H. Mino, I. Goto, G. Neubig, S. Kurohashi, and E. Sumita. Overview of the 2nd Workshop on Asian Translation. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 1–28, Kyoto, Japan, October 2015.
- [68] G. Neubig. Travatar: A Forest-to-String Machine Translation Engine Based on Tree Transducers. In *ACL (Conference System Demonstrations)*, pages 91–96. The Association for Computational Linguistics, 2013.
- [69] G. Neubig and K. Duh. On the Elements of an Accurate Tree-to-String Machine Translation System. In *The 52nd Annual Meeting of the Association for Computational Linguistics (ACL) Short Paper Track*, Baltimore, USA, June 2014.
- [70] J. Niehues, T. Herrmann, S. Vogel, and A. Waibel. Wider Context by Using Bilingual Language Models in Machine Translation. In *Proceedings of the*

- Sixth Workshop on Statistical Machine Translation*, pages 198–206, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.
- [71] F. J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. A. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. A Smorgasbord of Features for Statistical Machine Translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2004.
- [72] F. J. Och and H. Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, 2003.
- [73] Y. Oda, G. Neubig, S. Sakti, T. Toda, and S. Nakamura. Ckylark: A More Robust PCFG-LA Parser. In *Proceedings of NAACL 2015: Demo Track*, 2015.
- [74] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318. Association for Computational Linguistics, 2002.
- [75] S. Petrov, L. Barrett, R. Thibaux, and D. Klein. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.
- [76] J. R. Pierce and J. B. Carroll. *Language and Machines: Computers in Translation and Linguistics*. National Academy of Sciences/National Research Council, Washington, DC, USA, 1966.
- [77] M. Post and D. Gildea. Parsers as Language Models for Statistical Machine Translation. In *Proceedings of AMTA*, 2008.

- [78] C. Quirk, A. Menezes, and C. Cherry. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279, 2005.
- [79] J. Richardson, F. Cromières, T. Nakazawa, and S. Kurohashi. KyotoEBMT: An Example-Based Dependency-to-Dependency Translation Framework. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.
- [80] J. Richardson, T. Kudo, H. Kazawa, and S. Kurohashi. A Generalized Dependency Tree Language Model for SMT. *Journal of Natural Language Processing*, 23(3), 2016.
- [81] J. Riesa, A. Irvine, and D. Marcu. Feature-Rich Language-Independent Syntax-Based Alignment for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 497–507. Association for Computational Linguistics, 2011.
- [82] R. Rosa, D. MareÅłdek, and A. Tamchyna. Deepfix: Statistical Post-Editing of Statistical Machine Translation Using Deep Syntactic Analysis. In *Proceedings of the Student Research Workshop at the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.
- [83] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, pages 65–386, 1958.
- [84] L. Schwartz, C. Callison-Burch, W. Schuler, and S. Wu. Incremental Syntactic Language Models for Phrase-Based Translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 620–631, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

- [85] H. Schwenk. Continuous Space Language Models. *Computer Speech and Language*, 21(3):492–518, July 2007.
- [86] R. Sennrich. Modelling and Optimizing on Syntactic N-Grams for Statistical Machine Translation. *Transactions of the Association for Computational Linguistics*, 3:169–182, 2015.
- [87] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 1948.
- [88] L. Shen, J. Xu, and R. M. Weischedel. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Association for Computational Linguistics*, 2008.
- [89] M. Shen, D. Kawahara, and S. Kurohashi. A Reranking Approach for Dependency Parsing with Variable-sized Subtree Features. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 308–317, Bali, Indonesia, November 2012. Faculty of Computer Science, Universitas Indonesia.
- [90] Y. Shen, C. Chu, F. Cromierès, and S. Kurohashi. Cross-Language Projection of Dependency Trees for Tree-to-tree Machine Translation. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation, PACLIC 29*, 2015.
- [91] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernández. Syntactic Dependency-Based N-grams As Classification Features. In *Proceedings of the 11th Mexican International Conference on Advances in Computational Intelligence, MICAI’12*, pages 1–11. Springer-Verlag, 2013.
- [92] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the

- Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529:484–503, 2016.
- [93] M. Simard, C. Goutte, and P. Isabelle. Statistical Phrase-Based Post-Editing. In *Proceedings of HLT-NAACL*, 2007.
- [94] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.
- [95] K. Spreyer and J. Kuhn. Data-Driven Dependency Parsing of New Languages Using Incomplete and Noisy Training Data. In *Proceedings of CoNLL*, pages 12–20, Boulder, CO, USA, June 2009.
- [96] M. Sundermeyer, R. Schlüter, and H. Ney. LSTM Neural Networks for Language Modeling. In *Interspeech*, pages 194–197, Portland, OR, USA, 2012.
- [97] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.
- [98] L. Tesnière. *Éléments de Syntaxe Structurale*. Klincksieck, Paris, France, 1959.
- [99] C. Tillmann. A Unigram Orientation Model for Statistical Machine Translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 101–104, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.
- [100] K. Toutanova, H. Suzuki, and A. Ruopp. Applying Morphology Generation Models to Machine Translation. In *Proceedings of ACL-08: HLT*, pages 514–522, Columbus, OH, USA, June 2008. Association for Computational Linguistics.

- [101] J. Uszkoreit, J. M. Ponte, A. C. Popat, and M. Dubiner. Large Scale Parallel Document Mining for Machine Translation. In *Proceedings of the 23rd International Conference on Computational Linguistics, COLING '10*, pages 1101–1109, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [102] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, USA, 1995.
- [103] S. Vogel, H. Ney, and C. Tillmann. HMM-Based Word Alignment in Statistical Translation. In *Proceedings of the 16th Conference on Computational Linguistics, COLING '96*, pages 836–841, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.
- [104] X. Wu, T. Matsuzaki, and J. Tsujii. Effective Use of Function Words for Rule Generalization in Forest-Based Translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31, Portland, OR, USA, June 2011. Association for Computational Linguistics.
- [105] F. Xia and M. McCord. Improving a Statistical MT System with Automatically Learned Rewrite Patterns. In *Proceedings of the 20th International Conference on Computational Linguistics, COLING '04*, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.
- [106] D. Yarowsky, G. Ngai, and R. Wicentowski. Inducing Multilingual Text Analysis Tools via Robust Projection Across Aligned Corpora. In *Proceedings of the First International Conference on Human Language Technology Research, HLT '01*, pages 1–8, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.
- [107] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701, 2012.

- [108] M. Zhang, H. Jiang, A. Aw, J. Sun, S. Li, and C. L. Tan. A Tree-to-Tree Alignment-Based Model for Statistical Machine Translation. In *Proceedings of the MT Summit XI*, 2007.

List of Major Publications

- [1] John Richardson, Taku Kudo and Hideto Kazawa. Projected Dependency Language Models for Syntactic Post-Editing without Target-Side Parsers. In *Proceedings of the 26th International Conference on Computational Linguistics* (2016) [submitted manuscript].
- [2] Toshiaki Nakazawa, John Richardson and Sadao Kurohashi. Insertion Position Detection Model for Flexible Non-Terminals. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (2016) [to appear].
- [3] John Richardson, Taku Kudo, Hideto Kazawa and Sadao Kurohashi. A Generalized Dependency Tree Language Model for SMT. *Journal of Natural Language Processing*, Vol. 23, No. 3. (2016).
- [4] John Richardson, Fabien Cromieres, Toshiaki Nakazawa and Sadao Kurohashi. Flexible Non-Terminals for Dependency Tree-to-Tree Reordering. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2016).
- [5] John Richardson, Toshiaki Nakazawa and Sadao Kurohashi. Enhancing Function Word Translation with Syntax-Based Statistical Post-Editing. In *Proceedings of the 6th Workshop on Patent and Scientific Literature Translation* (2015).
- [6] John Richardson, Fabien Cromieres, Toshiaki Nakazawa and Sadao Kurohashi. KyotoEBMT System Description for the 2nd Workshop on Asian Translation. In *Proceedings of the 2nd Workshop on Asian Translation* (2015).
- [7] John Richardson, Fabien Cromieres, Toshiaki Nakazawa and Sadao Kurohashi. KyotoEBMT: An Example-Based Dependency-to-Dependency Translation Framework (System Demonstration). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (2014).

- [8] John Richardson, Fabien Cromieres, Toshiaki Nakazawa and Sadao Kurohashi. KyotoEBMT System Description for the 1st Workshop on Asian Translation. In *Proceedings of the 1st Workshop on Asian Translation* (2014).

List of Other Publications

- [1] Shinsuke Mori, John Richardson, Tetsuro Sasada, Hiroataka Kameko and Yoshimasa Tsuruoka. A Japanese Chess Commentary Corpus. In *Proceedings of the 10th Conference on International Language Resources and Evaluation* (2016).
- [2] John Richardson, Toshiaki Nakazawa and Sadao Kurohashi. Pivot-Based Topic Models for Low-Resource Lexicon Extraction. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation* (2015).
- [3] Koichi Akabe, Graham Neubig, Taku Kudo, John Richardson, Toshiaki Nakazawa and Sho Hoshino. Machine Translation Error Analysis for Project Next (in Japanese). In *Proceedings of the 20th Annual Meeting of the Association for Natural Language Processing* (2015).
- [4] John Richardson, Toshiaki Nakazawa and Sadao Kurohashi. Bilingual Dictionary Construction with Transliteration Filtering. In *Proceedings of the 9th Conference on International Language Resources and Evaluation* (2014).
- [5] John Richardson, Toshiaki Nakazawa and Sadao Kurohashi. Pivot, Box and Trilingual: Lexicon Extraction for Low-Resource Language Pairs using Extended Topic Models. In *Proceedings of the 20th Annual Meeting of the Association for Natural Language Processing* (2014).
- [6] Yuta Tsuboi, Hiroshi Kanayama, Katsumasa Yoshikawa, Tetsuya Nasukawa, Akihiro Nakayama, Kei Kanno and John Richardson. Converting Dependency Parsers from Rule-Based to Learning-Based (in Japanese). In *Proceedings of the 20th Annual Meeting of the Association for Natural Language Processing* (2014).
- [7] John Richardson, Toshiaki Nakazawa and Sadao Kurohashi. Robust Transliteration Mining from Comparable Corpora with Bilingual Topic Models. In

Proceedings of the 6th International Joint Conference on Natural Language Processing (2013).