Deep Syntax in Statistical Machine Translation

Yvette Graham

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

Doctor of Philosophy (Ph.D.)

to the

Dublin City University
School of Computing

Supervisor: Prof. Josef van Genabith

January 2011

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

Signed: Yvette Graham

(Candidate) ID No.: 98455842

Date: 19th January 2011

Abstract

Statistical Machine Translation (SMT) via deep syntactic transfer employs a three-stage architecture, (i) parse source language (SL) input, (ii) transfer SL deep syntactic structure to the target language (TL), and (iii) generate a TL translation. The deep syntactic transfer architecture achieves a high level of language pair independence compared to other Machine Translation (MT) approaches, as translation is carried out at the more language independent deep syntactic representation. TL word order can be generated independently of SL word order and therefore no re-ordering model between source and target words is required. In addition, words in dependency relations are adjacent in the deep syntactic structure, allowing the extraction of more general transfer rules, compared to other rules/phrases extracted from the surface form corpus, as such words are often distant in surface form strings, as well as allowing the use of a TL deep syntax language model, which models a deeper notion of fluency than a string-based language model and may lead to better lexical choice. The deep syntactic representation also contains words in lemma form with morpho-syntactic information, and this enables new inflections of lemmas not observed in bilingual training data, that are out of coverage for other SMT approaches, to fall within coverage of deep syntactic transfer.

In this thesis, we adapt existing methods already successful in Phrase-Based SMT (PB-SMT) to deep syntactic transfer as well as presenting new methods of our own. We present a new definition for consistent deep syntax transfer rules, inspired by the definition for a consistent phrase in PB-SMT, and we extract all rules consistent with the node alignment, as smaller rules provide high coverage of unseen data, while larger rules provide more fluent combinations of TL words. Since large numbers of consistent transfer rules exist per sentence pair, we also provide an efficient method of extracting rules as well as an efficient method of storing them. We also present a deep syntax translation model, as in other SMT approaches, we use a log-linear combination of features functions, and include a translation model computed from relative frequencies of transfer rules, lexical weighting, as well as a deep syntax language model and string-based language model. In addition, we describe methods of carrying out transfer decoding, the search for TL deep syntactic structures, and how we efficiently integrate a deep syntax trigram language model to decoding, as well as methods of translating morpho-syntactic information separately from lemmas, using an adaptation of Factored Models.

Finally, we include an experimental evaluation, in which we compare MT output for different configurations of our SMT via deep syntactic transfer system. We investigate various methods of word alignment, methods of translating morpho-syntactic information, limits on transfer rule size, different beam sizes during transfer decoding, generating from different sized lists of TL decoder output structures, as well as deterministic versus non-deterministic generation. We also include an evaluation of the deep syntax language model in isolation to the MT system and compare it to a string-based language model. Finally, we compare the performance and types of translations our system produces with a state-of-the-art phrase-based statistical

machine translation system and although the deep syntax system in general currently under-performs, it does achieve state-of-the-art performance for translation of a specific syntactic construction, the compound noun, and for translations within coverage of the TL precision grammar used for generation.

We provide the software for transfer rule extraction, as well as the transfer decoder, as open source tools to assist future research.

Acknowledgements

Firstly, I'd like to thank my supervisor, Josef, for giving me the chance to work on the project. I don't think I know anyone more hard-working and dedicated to the job than Josef.

A special thanks to Sue Fagan, Mary Chawke, Sinead McHugh, Seana Fagan and Karen Gibson, who are always there for me at the drop of a hat. Also, thank you to Grace and Stephen. Thanks too to Jennifer Foster, Deirdre Hogan, Sara Morrissey, Riona Finn, Joachim Wager, Ozlem Cetinoglu, Mary Hearne and Sisay Adafre. A big thank you to the rest of the group, Maria, Ankit, Djame, Ines, Tsuyoshi, Anton, Sergio, Wei, Garrath, Lamia, Mohammed, Johannes, ..., there are too many of you to mention everyone.

I'd also like to thank the members of LFG Pargram group. A special thanks to Martin Forst, Sina Zarriess, Tracy King, Sebastian Sulger, Nazareth Amlesom Kifle, John Maxwell and Ron Kaplan for their interest in and feedback on my work.

Another special thanks to the MT marathon participants and organisers, especially to Barry Haddow and Omar Zaidan, Philipp Koehn and Ondrej Bojar. A special thank you also to Carl Vogel and Donal Fitzpatrick, for their support and advice.

I'd really like to thank each of my examiners for taking time out of their busy schedules to examine my work.

I would also like to thank my family, my mum, Eileen, my sister Rosemary, and brothers, David and Martin, who have looked after me so well all my life including the past couple of years. Finally, I have to give a big thank you to my husband, Dermot, who has been so understanding and supportive over the past few years with everything.

> The glory of friendship is not the outstretched hand, not the kindly smile, not the joy of companionship; it is the ... inspiration that comes to one when he discovers that someone believes in him ...
>
> Ralph Waldo Emerson

Contents

List of Figures

Chapter 1

Introduction

1.1 Natural Language Processing

Early records provide evidence that humans have been analyzing language since as early as the 2^{nd} millennium B.C. The following extract from the *Old Babylonia Grammatical Texts* shows that translation motivated some of the earliest research into the science of language (Gragg, 1995):

Sumerian	Akkadian	English Gloss
ŋen-na	*alik*	'go!'
ga-ŋen	*lullik*	'may I go!'
he e-ŋen	*lillik*	'may he go!'
an-du	*illak*	'he goes'
an-du-un	*allak*	'I go'
an-du-un	*tallak*	'you go'

Today, although we no longer etch our analysis onto a stone tablet, but are more likely to type it on a keyboard or write a program to extract it from some electronically stored text, not all that much has changed in 4,000 or so years. Natural language still fascinates people and probably always will do. Everyone from all walks of life at some stage thinks about language and question how and why we use language the way we do. Language scientists have managed to provide answers to some but not all of these questions. Natural language engineering/processing (NLP) in particular can provide some insight into how language works (Uszkoreit, 2009).

Something that puzzles people that do not know much about NLP is why NLP is difficult at all. They expect a computer to be smarter than they are themselves and cannot understand why something they find so ridiculously simple could be difficult for a computer. The reason they believe this, besides over-estimating the abilities of a computer, is that they underestimate their own intelligence. The inherent tendency to only compare things that are relatively similar to each other, causes people to take for granted their amazingly intelligent ability to communicate through language. They believe speaking or writing with prescriptively grammatical language is a sign of intelligence. However, how human beings use natural language to communicate at all is the significantly intelligent part. Focusing on something like prescriptive grammar, in the grand scheme of things, is like worrying about getting the flag straight when you've already made it to the moon.

Most NLP tasks are not easily solved because language itself is so complex, and NLP provides an excellent test for the degree to which language scientists really understand their topic. Like with any puzzle, solving it provides pretty good evidence that you really understand it.

1.2 Machine Translation

One popular NLP challenge is Machine Translation (MT), successfully automating the process of translation from one natural language to another. What fascinated Babylonians about language is partly what makes MT so fascinating: the diversity of language. By comparing text translated into another language, we can gain a great deal of insight into how language in general works, and as a test for the degree to which language scientists really understand language, the task of MT is probably the best test of all. To produce a machine that can translate between any pair of languages in the world has to be both one of the most interesting challenging tasks in NLP.

1.3 Deep Syntax and Machine Translation

Essentially, MT systems need to accomplish two things: translate the SL words into the TL and produce these words in the correct order for the TL (Koehn, 2009). Approaches to MT use different levels of linguistic analysis for translation and divide the tasks involved in the translation of words and word order between analysis and generation (AG) components and a transfer (T) component, as shown in Figure 1.1. The shallowest approach translates a SL surface form sentence directly

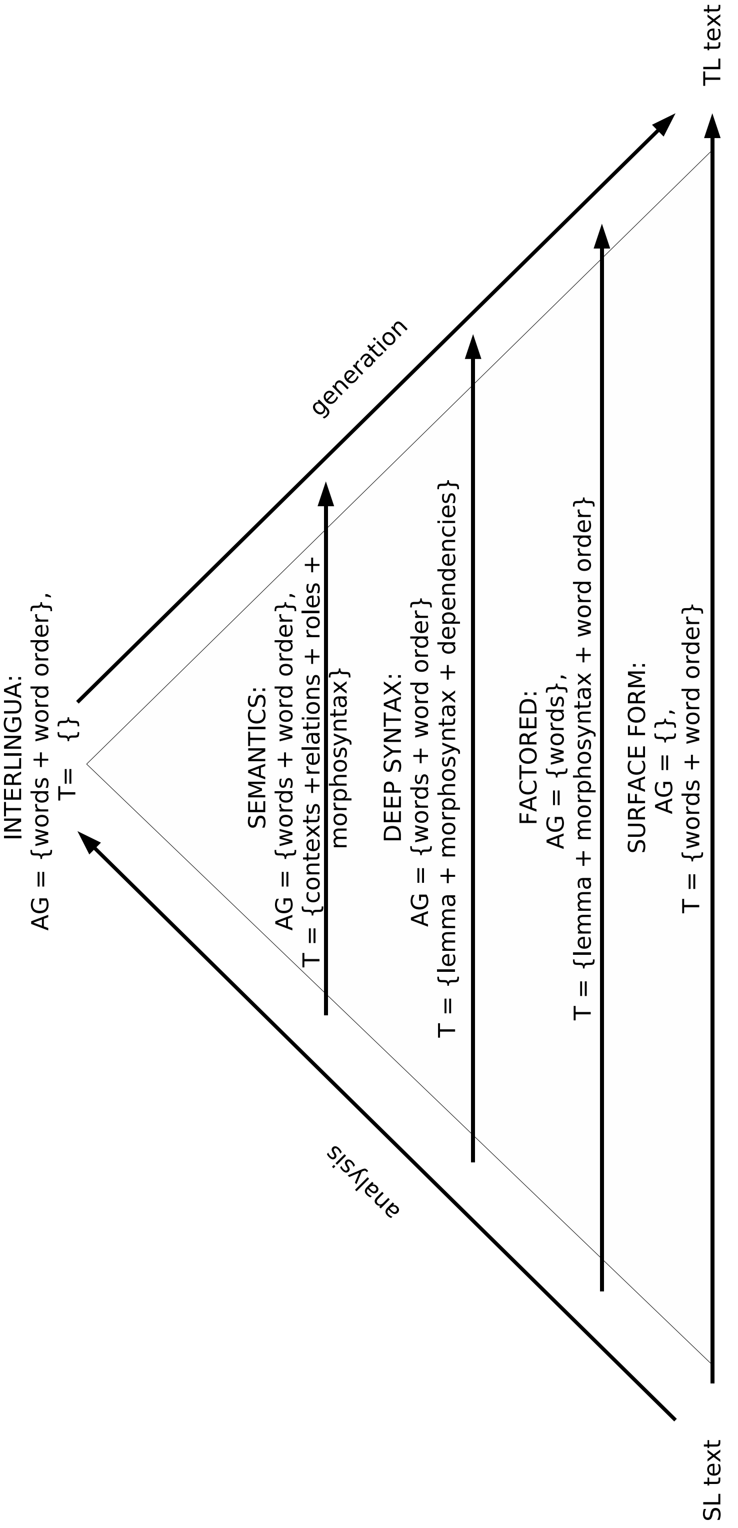

INTERLINGUA:
AG = {words + word order},
T= {}
generation
analysis
SEMANTICS:
AG = {words + word order},
T = {contexts +relations + roles + morphosyntax}
DEEP SYNTAX:
AG = {words + word order}
T = {lemma + morphosyntax + dependencies}
FACTORED:
AG = {words},
T = {lemma + morphosyntax + word order}
SURFACE FORM:
AG = {},
T = {words + word order}
SL text
TL text

Figure 1.1: Translation Pyramid

into the TL, assigning the tasks of translating both words and word order to the transfer component, as in Phrase-Based Statistical Machine Translation (PB-SMT) for example. At a slightly deeper level of analysis, such as Phrase-Based Factored Models, transfer involves translating the lemma form, morpho-syntactic information and word order to the TL. Deep syntactic analysis goes a level deeper and transfer now involves translating SL syntactic representations such as dependency relations, lemmas and morpho-syntactic information to the TL. Even deeper again we have semantic analysis, with transfer translating between SL and TL context and meaning representations, relations, roles and (possibly) morpho-syntactic information. Finally, an interlingual analysis assigns the entire translation task to the analysis and generation components, with no transfer required, since the representation itself is entirely language independent.

Although increasing the depth of analysis can potentially *decrease the difficulty of translation*, there is a trade-off as a deeper analysis *increases the difficulty of analysis and generation*. In addition, when we divide the task of translation into separate components in a pipeline architecture, we need to consider how well each step in the pipeline fits together. The output of the parser used for analysis must be the input expected by the transfer decoder, and likewise the transfer decoder output must provide good input for generation. In addition, the use of parsers and generators to a deep level of analysis can also restrict the number of translation hypotheses reached by the search. For example, if generation is only possible on the sentence level, as opposed to the word level, significantly more pruning of translation options may be necessary.

In theory, a deep syntactic analysis provides a good level of linguistic analysis for machine translation, for several reasons. Firstly, achieving the correct word order in the TL is one of the biggest challenges in MT and trying to devise a language pair independent way of reordering words between the source and target language is extremely difficult (Koehn, 2009; Crego and Habash, 2008; Crego and Marino, 2007; Dreyer et al., 2007; Chen et al., 2006; Costa-jussa and Fonollosa, 2006; Crego and

Marino, 2006). The rules that govern word order in a single language are already very complex, and coming up with a way of correctly translating the word order for *any* language pair is extremely difficult. If deep syntax is used as the level of analysis for MT, generation of TL word order can be carried out independently of SL word order. Other favourable properties of deep syntactic transfer include:

- Morpho-syntactic information for source and target sentences is present in deep syntactic representations, so Factored Models (Koehn and Hoang, 2007) can be used to provide statistically richer translation models and coverage of inflections of lemmas not observed in bilingual training data.

- The deep syntactic representation encodes dependency relations between words, which can help to produce more general rules/phrases for translation, as such words are often distant in the string, as well as enabling the integration of a TL deep syntax language model, that models a deeper notion of fluency than a string-based model.

- The availability of SL morpho-syntactic information can improve the translation from morphologically poor languages into morphologically richer languages (Avramidis and Koehn, 2008) and can potentially improve translation of specific forms of verbs that cause difficulty for other approaches, such as for example gerund verbs (Aranberri-Monasterio and O'Brien, 2009), as such information is explicitely present in the deep syntactic representation.

- The number of nodes in a deep syntactic representation is in general less than the number of words in the sentence, avoiding some of the complexity problems encountered by shallow-syntax based approaches (Deneefe and Knight., 2009; Deneefe et al., 2007; Charniak et al., 2003).

- Non-terminals are allowed in transfer rules to map pieces of SL structure to the correct position in the TL but in a much more constrained way than in, for example, Phrase-Based Hierarchical Models (Chiang, 2007b,a) avoiding the severe pruning necessary for decoding in such parsing-based approaches (Li et al., 2009; Chiang, 2007b,a).

- Decoding can be carried out via a top-down application of contiguous transfer rules, so there are no gaps between TL words, eliminating the need for sophisticated heuristic language modeling techniques as in Hierarchical Phrase-Based Models (Chiang, 2007b).

Some practical challenges still need to be overcome before state-of-the-art performance can be achieved with SMT via deep syntax, however. One challenge is parser coverage: depending on the parsing technologies used, coverage of long sentences can be very low, resulting in a much smaller sized bilingual corpus used for training in comparison with other approaches. A similar challenge occurs for generator coverage: technologies for generation from deep syntactic structures are usually tested on gold-standard input, and even with adaptation to allow more robust generation, generator coverage can still be low. Possibly the most significant challenge, however, is constructing good quality TL structures. For a single TL deep syntactic structure, the number of possible combinations of lemmas, dependency relations and morpho-syntactic information, is very high and automatically finding a single good combination is extremely challenging. Since we are restricted to sentence level generation, we are forced to severely prune translation options prior to generation and this greatly increases the likelihood of many good translations never being generated.

1.4 Research Questions & Motivations

The main research questions and motivation for the work included in this thesis are as follows:

- investigate the main challenges of using deep syntax for transfer in machine translation,

- apply machine learning methods: develop methods of fully automatic training,

- use a language pair independent approach,
- use a linguistic theory independent approach (in the sense that the approach can be easily applied to another theory of deep syntax),
- apply Phrase-Based SMT methods to deep syntactic transfer,
- develop efficient methods of training and decoding,
- make as many of the tools as possible open source to aid future research,
- investigate effects of system parameters on translation quality,
- provide an empirical comparison of deep syntactic transfer and Phrase-Based SMT.

1.5 Thesis Structure

We begin in Chapter 2, with the important background information for the thesis, such as the theory of deep syntax we use, LFG f-structures and the LFG parser/generator we use for experiments. In addition, we describe important methods in PB-SMT, in particular those we apply to deep syntax. We also critically review related work, providing detail about how our own approach uses similar or different methods.

In Chapter 3, we describe the deep syntax transfer rules we use in our MT system and a method of automatically extracting them from parsed bilingual corpora. We extract large numbers of transfer rules that each contain a lot of information. We provide a method for efficiently extracting and then storing large numbers of transfer rules.

In Chapter 4, we describe our translation model, a log-linear combination of feature functions including decoding features, such as a translation model computed from relative frequencies of extracted transfer rules, as well as a deep syntax language model. We also combine post-generation features in our model, such as a string-based language model and a grammaticality feature using information produced by the TL precision grammar about the grammaticality of an output translation.

In Chapter 5, we describe how transfer decoding is carried out via a heuristic search using the translation model described in Chapter 4 to rank translation hypotheses. We include detail of how we efficiently integrate a deep syntax trigram language model into decoding as well as how we use an adaptation of Factored Models (Koehn and Hoang, 2007) to translate morpho-syntactic information.

In Chapter 6 we provide a detailed evaluation of several configurations of our SMT via deep syntactic transfer system. We investigate the effects of different methods of word alignment, different methods of translating morpho-syntactic information and different limits on transfer rules size, in addition to different transfer decoder beam sizes, generating from different sized transfer decoder output lists, as well as investigating deterministic versus non-deterministic generation. We restrict our evaluation to a limited sentence length (5-15 words) for German to English translation mainly due to current parser and generator coverage and robustness limits. In addition, we include a comparison with state-of-the-art PB-SMT that shows although overall our system under-performs, that for the translation of a specific syntactic construction, the compound noun, our system achieves state-of-the-art performance. In addition, we show that the system achieves state-of-the-art performance for translations within coverage of the TL precision grammar used for generation. We also provide an evaluation of the deep syntax language model independently of the MT system, which we believe has the potential to improve lexical choice if integrated into Phrase-Based SMT systems, and compare it to a string-based language model. Finally, in Chapter 7, we provide some conclusions and possibilities for future work.

1.5.1 Thesis Contributions

The main contributions of this thesis are as follows:

- a new definition for consistent deep syntax transfer rules;
- a new method of automatically word/node aligning deep syntactic structures;
- a new method of extracting and storing deep syntax transfer rules (provided as open source software);
- a definition for deep syntax language modeling;
- efficient methods of incorporating deep syntax language modeling into transfer decoding (provided as open source software);
- a new method of translating morpho-syntactic information for SMT;
- a detailed experimental investigation into the effects of using different configurations in SMT via deep syntactic transfer.

Chapter 2

Deep Syntax and Phrase-Based SMT

2.1 Introduction

This thesis investigates applying standard methods of PB-SMT to deep syntactic transfer. This chapter describes the background to this approach and previous work in the area. Firstly, we outline the linguistic theory underlying the deep syntactic representations we use as the intermediate representation for transfer, the LFG f-structure, as well as the tools used to automatically (i) parse text to this representation and (ii) generate text from this representation. Following that, we present a detailed description of PB-SMT, before finally describing relevant previous work that combine SMT techniques and deep syntactic transfer.

2.2 Lexical Functional Grammar

LFG (Kaplan and Bresnan, 1982; Kaplan, 1995; Bresnan, 2001; Dalrymple, 2001) is a deep unification or constraint-based grammar formalism that minimally defines two levels of syntactic representation: constituent structure (c-structure) and functional structure (f-structure). LFGs for particular languages are specified by means of a grammar and lexicon. Figure 2.1 shows example LFG grammar rules and lexical entries, as well as, the c-structure and f-structure for the English sentence *"John loves Mary"*. The c-structure of a sentence encodes the string and its associated phrase structure tree, while the f-structure encodes the corresponding predicate-argument (or dependency) structure as an attribute value matrix. For a given sentence, a functional projection, ϕ, maps each node in the c-structure to a f-structure, shown in Figure 2.1 as arrows from each c-structure node to a local f-structure.

A LFG grammar consists of context free grammar rules with added constraints, for example in Figure 2.1 the grammar rule for S expands to a NP with constraint $(\uparrow \text{SUBJ}) =\downarrow$ and VP with constraint $\uparrow=\downarrow$. A LFG lexicon consists of lexical entries each of which contains the surface form word with its pre-terminal symbol and a set of constraints, for example the lexical entry for *John* has the pre-terminal category NP and constraints $(\uparrow \text{PRED}) =$'John', $(\uparrow \text{NUM}) =$sg, $(\uparrow \text{PERS}) =3$ and $(\uparrow \text{GEND})$

=masc. The notation used for specifying grammar and lexicon constraints includes the metavariables ↓ and ↑. The metavariable ↓ denotes the ϕ-image (effectively the f-structure) of the c-structure node to which the constraint is attached, whereas ↑ denotes the ϕ-image (the f-structure) of the mother node in the c-structure of the c-structure node to which the constraint is attached. For example, in the grammar rule for VP in Figure 2.1, the constraint attached to the V, ↑=↓, means *the mother of V (i.e. VP) has the same f-structure as V* and the constraint on the NP of the same grammar rule, (↑ OBJ) =↓, means *the* object *of the f-structure of the mother of NP (i.e. the object of the f-structure of VP) is the f-structure of NP*.

Parsing a sentence to f-structure involves the unification of constraints attached to the grammar rules that are used to parse the sentence and constraints attached to the lexical entries of the words in the sentence. For example, in Figure 2.1 the constraint on the lexical entry for *loves*, (↑ SUBJ PERS)=3, and the constraint on the lexical entry for *John*, (↑ PERS) =3, unify with each other effectively enforcing subject-verb agreement.

Once a sentence is parsed, its f-structure is likely to contain several f-structures nested within each other. The term *local f-structure* is used when referring to individual f-structures contained within the (outermost) f-structure, inclusive of the (outermost) f-structure itself. The attributes of a local f-structure, that were defined by the constraints of the grammar and lexicon such as *SUBJ*, *OBJ*, *TENSE* etc., form an unordered set, each attribute having exactly one value and being either complex, i.e. its value is another f-structure, or atomic, i.e. it has an atomic value. The complex attributes of the local f-structures encode the underlying abstract syntactic functions between each predicate of the sentence and its arguments, such as SUBJ, OBJ, COMP (complement), XCOMP (x-complement) and OBL (oblique), whereas atomic attributes encode other information that play a role in the functional syntax, such as PERS, GEND, NUM, CASE and TENSE. An f-structure can also contain the ADJ (adjunct) function, the value of which is *a set of f-structures.* The PRED (predicate) attribute of a local f-structure has a semantic value, a reference to an entry containing the lexical semantics of the item and consisting of the lemma form of the word and a list of the grammatical functions of its arguments. The term *preds-only* f-structure is used to refer to an f-structure without its atomic features and values, i.e in a preds-only f-structure each local f-structure only contains the PRED feature and value and the complex features and values. Figure 2.2 shows the preds-only f-structure of English sentence *"John loves Mary"*.

2.2.1 XLE

In our work to date, we have used the Xerox Linguistic Environment (XLE) parse engine and generator (Maxwell and Kaplan, 1993, 1996; Kaplan and Maxwell, 1996; Kaplan et al., 2002) for LFG parsing and generation. XLE uses a number of components to carry out parsing and generation including an LFG grammar, LFG lexicon, a tokenizer and a finite state morphological analyser. Figure 2.3 shows the order in which these components are applied when text is parsed with XLE. For generation the same components are applied in reverse order to an input f-structure.

XLE produces all possible parses according to the grammar and lexicon for a sentence in a packed representation. If the single most probable parse is required, XLE includes a search algorithm for searching for the best parse according to a stochastic disambiguation model, see for example Forst (2007) for German parse disambiguation and Riezler et al. (2002); Kaplan et al. (2004) for English parse disambiguation. The disambiguation model is defined as a log-linear combination of over 1000 feature functions comprising information about c-structure, f-structure and lexical elements. Dynamic programming is used for efficient search for the most probable parse from the packed parse representation. A threshold limit on the amount of work done when evaluating the features functions is used to limit the time spent searching a parse forest. If the threshold is reached, no more feature values are computed and the most probable parse is simply selected using the set of features that have been evaluated thus far.

For generation, for a single f-structure there often exists more than one possi-

LFG Grammar Rules

$$\begin{array}{lccc}
\mathrm{S} & \rightarrow & \mathrm{NP} & \mathrm{VP} \\
 & & (\uparrow \mathrm{SUBJ}) =\downarrow & \uparrow=\downarrow \\
 & & & \\
\mathrm{VP} & \rightarrow & \mathrm{V} & \mathrm{NP} \\
 & & \uparrow=\downarrow & (\uparrow \mathrm{OBJ}) =\downarrow
\end{array}$$

Lexical Entries

John	NP	(↑ PRED) =‘John’
		(↑ NUM) =sg
		(↑ PERS) =3
		(↑ GEND) =masc
loves	V	(↑ PRED)=‘love⟨SUBJ, OBJ⟩’
		(↑ TENSE)=present
		(↑ SUBJ PERS)=3
		(↑ SUBJ NUM)=sg
Mary	NP	(↑ PRED)=‘Mary’
		(↑ NUM)=sg
		(↑ PERS)=3
		(↑ GEND)=fem

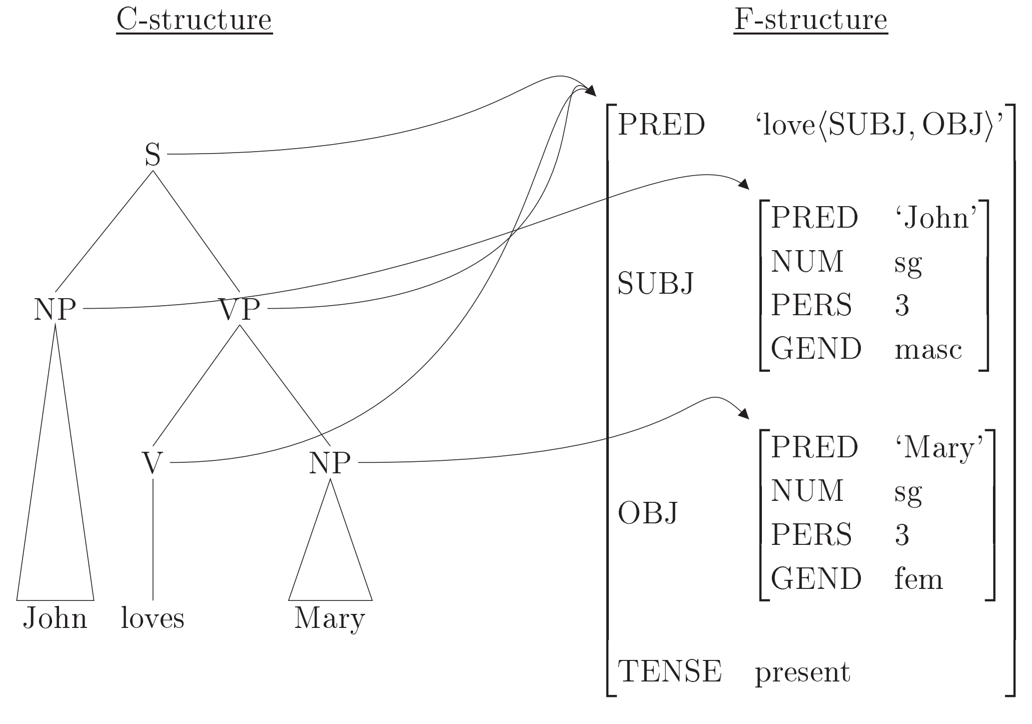


Figure 2.1: LFG grammar rules and lexical entries with C-structure and F-structure for the English sentence *“John loves Mary.”*

$$\left[\begin{array}{ll} \mathrm{PRED} & \text{‘love}\langle\mathrm{SUBJ},\mathrm{OBJ}\rangle\text{’} \\ \\ \mathrm{SUBJ} & \left[\begin{array}{ll}\mathrm{PRED} & \text{‘John’}\end{array}\right] \\ \\ \mathrm{OBJ} & \left[\begin{array}{ll}\mathrm{PRED} & \text{‘Mary’}\end{array}\right] \\ \end{array}\right]$$

Figure 2.2: Preds-only f-structure for English sentence *“John loves Mary.”*

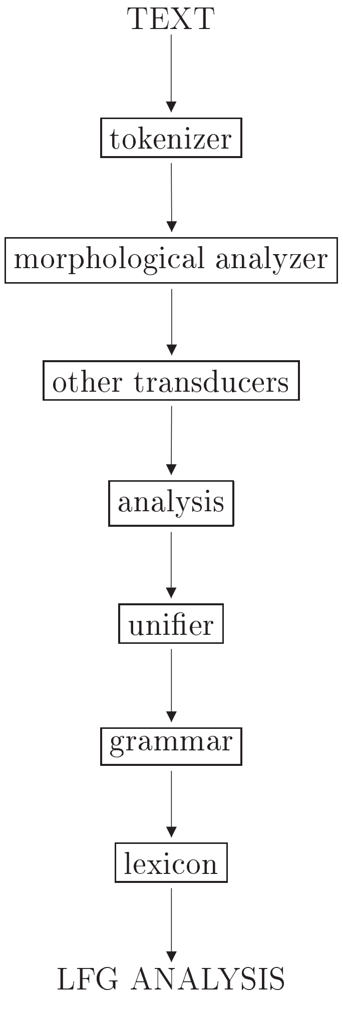


Figure 2.3: XLE Parser Components

ble surface form realisation. XLE includes an option to generate the surface form sentences of an f-structure in a packed representation or to simply enumerate them. The packed representation XLE uses however for generation is not efficient or compact enough in most cases to provide any real advantage over enumerating sentences separately when computing language model scores for sentences, such as, for example, the efficient forest structures of Langkilde (2000) used for statistical generation, in which alternate phrases are represented as packed sets of trees facilitating more efficient statistical ranking. For grammatical f-structures (f-structures that do not cause constraint clashes during generation) the number of possible outputs is usually manageable, so the enumeration option can be used without slowing down the overall system significantly. However, when input structures are ungrammatical, the number of generated surface form sentences can be in the millions, and lack of an efficient method of scoring such numbers of sentences is prohibitive. Due to time constraints, we leave the integration of more sophisticated methods of generation to future work, however. XLE has three options for generation, *longest*: deterministically producing the longest string for the input structure according to the grammar, *shortest*, producing the shortest string, and *allstrings*, producing all possible strings for the input TL structure according to the grammar. We refer to these options as *k-options*, and investigate the effect on MT output of using alternate k-options later in our evaluation in Chapter 6.

2.2.2 Statistical LFG Parsing and Generation Resources

Statistical LFG parsing resources are available for English[1], German[2], Chinese[3] and Spanish[4]. A probabilistic context free grammar parser (Petrov et al., 2006; Charniak and Johnson, 2005; Klein and Manning, 2003b,a) is first used to parse raw text to

[1] (Cahill et al., 2008; Chrupala and van Genabith, 2007; Chrupala et al., 2007; O'Donovan, 2006; Cahill et al., 2005; Judge et al., 2005; O'Donovan et al., 2005b,a; Cahill et al., 2004; Cahill, 2004; Burke et al., 2004a; O'Donovan et al., 2004; Cahill et al., 2002a,b)

[2] (Rehbein, 2009; O'Donovan, 2006)

[3] (Guo, 2009; Guo et al., 2007b,a; O'Donovan, 2006; Burke et al., 2004b)

[4] (Chrupala, 2008; Chrupala and van Genabith, 2006; O'Donovan, 2006)

Penn Treebank (Marcus et al., 1994, 1993) style phrase-structure trees, before an annotation algorithm is applied to the trees to produce f-structures. Compared to the XLE style f-structures, the statistical resources produce less fine-grained analyses that contain fewer atomic features/morpho-syntactic information. Statistical resources for generation have also been developed for English (Cahill and van Genabith, 2006; Hogan et al., 2007) and Chinese (Guo, 2009; Guo et al., 2008a,b).

In Graham et al. (2007), we compared the performance of these parsing and generation resources for English with XLE, by regenerating English Europarl sentences and effectively finding the upper bound imposed on a transfer based MT system that employs the particular parsing/generation technologies. Results showed the statistical resources achieve a higher upper bound for unrestricted sentence length (XLE: 47.85% BLEU; statistical resources: 57.16% BLEU), and conversely XLE achieves a higher upper bound for short sentences (XLE: 74.31% BLEU; statistical resources: 69.68% BLEU).

2.3 Phrase-Based Statistical Machine Translation

Given the availability of a large bitext corpus of any language pair, a Statistical Machine Translation (Brown et al., 1988, 1990) (SMT) system automatically learns how to translate unseen text from one language to another. SMT literature follows the convention of describing the source language as *the foreign language* and the target language as *English*, we follow this convention also. In SMT, a translation model is computed from the training corpus and when given an unseen foreign sentence, **f**, as input to the system, the model is used to estimate the probability of each candidate translation **e** given **f**, with the ultimate goal of finding the best English translation, $\hat{\mathbf{e}}$, for **f**:

$$\hat{\mathbf{e}} = argmax_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \tag{2.1}$$

SMT systems use language models to model fluent target language text. The noisy-channel model (Shannon, 1948), borrowed from speech recognition, applies Bayes Rule to $p(\mathbf{e}|\mathbf{f})$ to produce Equation 2.2 motivating the use of a language model.

$$argmax_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = argmax_{\mathbf{e}} \frac{p(\mathbf{f}|\mathbf{e})p(\mathbf{e})}{p(\mathbf{f})} \tag{2.2}$$

Since p($\mathbf{f}$) is a constant for all $\mathbf{e}$ it is redundant in computing $argmax_{\mathbf{e}} p(\mathbf{e}|\mathbf{f})$ and can be dropped from Equation 2.2 to obtain Equation 2.3.

$$argmax_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = argmax_{\mathbf{e}} p(\mathbf{f}|\mathbf{e})p(\mathbf{e}) \tag{2.3}$$

The noisy channel model also motivates the use of the reverse translation direction model. Besides the language model and reverse translation direction model, state-of-the-art SMT systems use several other components to compute $p(\mathbf{e}|\mathbf{f})$. Equation 2.4 shows how $p(\mathbf{e}|\mathbf{f})$ can be defined as a log-linear combination of several feature functions(Och and Ney., 2002).

$$p(e|f) = exp \sum_{i=1}^{n} \lambda_i h_i(e, f) \tag{2.4}$$

2.3.1 Translation Model

In PB-SMT the translation of a foreign sentence, $\mathbf{f}$, into an English sentence, $\mathbf{e}$, is modeled by breaking down the translation of the sentence into the translation of a set of phrases:

$$p(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)$$

To compute $\phi(\bar{f}|\bar{e})$, the bitext corpus is automatically word-aligned before all phrases consistent with the word alignment are extracted. The Maximum Likelihood Estimation (MLE) for $\phi(\bar{f}|\bar{e})$ is computed using Equation 2.5.

$$\phi(\bar{f}|\bar{e}) = \frac{count(\bar{e}, \bar{f})}{\sum_{\bar{f}_i} count(\bar{e}, \bar{f}_i)} \tag{2.5}$$

This is carried out for both language directions, so that the *direct translation direction* translation model and *reverse translation direction* translation model can be used as features.

2.3.2 Lexical Weighting

Lexical weighting is used as a back-off to the translation model as it provides richer statistics and more reliable probability estimates. The lexical translation probability of a phrase pair is computed using the alignment between the words in the phrase pair. The lexical translation probability of a phrase, $\bar{e}$, given the phrase $\bar{f}$ and alignment a, is estimated as follows:

$$lex(\bar{e}|\bar{f}, a) = \prod_{i=1}^{length(\bar{e})} \frac{1}{|\{j|(i,j) \in a\}|} \sum_{\forall (i,j) \in a} w(e_i|f_j)$$

Like the translation model, lexical weighting can be modeled in both language directions.

2.3.3 Language Model

Ngram language models are used in PB-SMT to help produce fluent output. Equation 2.6 shows how the probability of a sequence of English words can be computed by combining the probability of each word, w_i, in the sequence given the preceding sequence of $i-1$ words.

$$P(w_1, ..., w_m) = \prod_{i=1}^{m} P(w_i|w_1, ..., w_{i-1}) \tag{2.6}$$

Although language models are computed using a large training corpus, statistics for long histories of words are unreliable due to data sparseness (Koehn, 2009), and therefore the Markov assumption is applied to Equation 2.6, which simplifies the probability of a sequence of words by using a limited history length when calculating the probability of each word. Equation 4.3 shows how an ngram language model computes the probability of the i^{th} word by the probability of observing it preceeded by its n-1 preceding words.

$$P(w_1, ..., w_m) = \prod_{i=1}^{m} P(w_i | w_{i-n-1}, w_{i-n-2}, ..., w_{i-1}) \tag{2.7}$$

Language models are evaluated using the perplexity measure. Equation 2.8 shows the definition of perplexity which is based on the cross-entropy of the language model. The definition of cross-entropy of a language model is shown in Equation 2.9.

$$PP = 2^{H(p_{LM})} \tag{2.8}$$

$$H(p_{LM}) = -\frac{1}{m} \sum_{i=1}^{m} log p_{LM}(w_i | w_{i-n-1}, ..., w_{i-1}) \tag{2.9}$$

2.3.4 Word and Phrase Penalty

The language model is biased towards shorter output, since adding a word to a sentence introduces an extra ngram and therefore including it reduces the overall probability of the sentence. A word penalty feature is used in order to allow a system to counter-balance the effects of this bias towards shorter output.

In a similar way, a phrase penalty is used to allow the system to bias towards using short or long phrases. Longer phrases might be more reliable than shorter ones because longer phrases ensure that the sequence of words in the English side of the phrase is fluent, since it was previously observed in the training data, compared to a sequence of words constructed from several short phrases (Koehn, 2009).

2.3.5 Lexicalized Reordering

The word aligned training corpus is used to model reordering for each phrase pair using information about how the phrase pair that precedes it in the English text moved with respect to its position in the foreign text. The following three types of reordering are allowed in the lexicalized reordering model (Koehn, 2009):

- monotone: if the immediately preceding phrase in the English text corresponds to the immediately preceding phrase in the foreign text;

- swap: if the immediately preceding preceding phrase in the English text corresponds to the immediate subsequent phrase in the foreign text;
- discontinuous: if the immediately preceding phrase in the English text neither corresponds to the immediately preceding phrase nor the immediate subsequent phrase in the foreign text.

Equation 2.10 defines how the lexical reordering model is computed using MLE.

$$p_o(orientation|\bar{f}, \bar{e}) = \frac{count(orientation, \bar{f}, \bar{e})}{\sum_o count(o, \bar{f}, \bar{e}} \tag{2.10}$$

The simplicity of the reordering model in PB-SMT hints at the difficulty of the task. The rules that govern what is considered a grammatical sequence of words (or grammatical sentence) can differ dramatically from one language to the next. Even within a single language, the words of a grammatical sentence can often be reordered to form other grammatical sentences retaining the original meaning.

In PB-SMT, all possible ways of reordering words/phrases between a language pair are divided into three different types. Monotone translation and swap are the two main types and any other type of reordering is classified as discontinuous. Lots of different types of reordering may be legitimate between the words/phrases of two translations, but only monotone and swap are given their own type in PB-SMT, leaving no distinction between all other types of reordering.

A major advantage of the *statistical* MT approach is language pair independence. Coming up with a reliable way of modeling the reordering of words between a specific pair of languages is a difficult task in itself, but defining a language pair independent method of reordering is even more challenging. The problem is that translating between different pairs of languages will involve different types of reordering, but a language pair independent model for reordering must be able to learn what types of reordering happen for any language pair and when they can be applied.

A favourable property of SMT via deep syntactic transfer is that no reordering model between the source and target language is necessary, since translation happens between deep syntactic structures and not surface form sentences, TL word order is generated independently of SL word order.

2.3.6 Word Alignment

In SMT, the alignment function $a : j \rightarrow i$ is used to specify correspondences between a word e_j of an English sentence $\mathbf{e} = (e_1, ...e_{l_e})$ with a word f_i of a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$. The special NULL token is included as an extra word in the foreign sentence to provide output to the alignment function for English words that have no corresponding foreign words in the translation. Expectation Maximization (EM) is applied to the IBM Models (Brown et al., 1990, 1993) for automatic word alignment on a sentence-aligned bitext corpus and the Viterbi (most probable) alignment is used as input to phrase extraction. The Expectation Maximization Algorithm is as follows (Koehn, 2009):

1. Initialize the model.
2. Apply the model to the data.
3. Learn the model from the data.
4. Iterate Steps 2 and 3 until convergence.

The IBM Models increase in complexity by adding phenomena that occur between translations of two languages that are increasingly complex to model (Koehn, 2009):

- IBM Model 1: lexical translation;
- IBM Model 2: adds absolute alignment model;
- IBM Model 3: adds fertility model;
- IBM Model 4: adds relative alignment model;

- IBM Model 5: fixes deficiency.

Often between translations of two languages, not only one-to-one alignment occurs, but also one-to-many, for example *"Hausfrau"* and *"house wife"*, many-to-one, for example *"stieg um"* and *"changed"*, and many-to-many, for example *"Großbritannien und Nordirland"* and *"United Kingdom"*. Since the alignment function takes in an English word position and returns a foreign word position, when word alignment is run in the foreign-English direction, one foreign word can be aligned with multiple English words, i.e. the one-to-many alignment of *"Hausfrau"* and *"house wife"* is possible, as multiple English words are allowed to have the same output from the alignment function. For example, the output could include an alignment between both *"house"* and *"Hausfrau"* and *"wife"* and *"Hausfrau"*. However, multiple foreign words cannot be aligned to a single English word, so running word alignment in the foreign-to-English direction cannot output both an alignment between *"changed"* and *"stieg"* and *"changed"* and *"um"*. In order to capture such many-to-one types of alignment, automatic word alignment is run in the reverse translation direction also. The alignment for the direct translation direction can then be combined with the alignment for the reverse translation direction. In addition, by combining the bidirectional word alignment it's possible to attain many-to-many alignments. The most commonly used method of combining the bidirectional word alignment (a.k.a. symmetrization) is the grow-diag-final algorithm (Koehn et al., 2003) that starts with the intersection and iteratively adds additional alignment points that neighbour other alignment points and unaligned words.

2.3.7 Phrase Extraction

All phrases consistent with the word alignment are extracted. The definition of a consistent phrase (Och et al., 1999; Koehn et al., 2003) is as follows:

Definition 1. *A phrase pair* $(\bar{f}, \bar{e})$ *is consistent with an alignment, A, if all words* $f_1, ..., f_n$ *in* $\bar{f}$ *that have alignment points in A, have these with words* $e_1, ..., e_n$ *in* $\bar{e}$

and vice versa:

$$
\begin{aligned}
(\bar{f}, \bar{e}) \textit{ consistent with } A \Leftrightarrow & \\
& \forall e_i \in \bar{e} : (e_i, f_j) \in A \Rightarrow f_j \in \bar{f} \\
\textit{and} \quad & \forall f_j \in \bar{f} : (e_i, f_j) \in A \Rightarrow e_i \in \bar{e} \\
\textit{and} \quad & \exists e_i \in \bar{e}, f_j \in \bar{f} : (e_i, f_j) \in A
\end{aligned}
$$

2.3.8 Decoding

Decoding a foreign sentence involves a search for the best translation according to the model. The translation process is carried out in sequence from left to right for the English output text, with reordering enabled by allowing the input phrases and their translations to have a different order.

Hypothesis Recombination

As translation hypotheses (partial translations) are built (from left to right in the output sequence) their probability is computed. During the search for the best translation according to the model, certain translation hypotheses will be encountered that cannot possibly form part of the highest scoring completed translation. For efficiency these translation hypotheses are dropped from the search. For example, if multiple translation hypotheses that cover the same part of the foreign sentence and produce the same English translation (but were produced by different sets of phrases) are encountered they are recombined by dropping the lower scoring hypotheses. In addition, if for multiple translation hypotheses the last n-1 words of the English output are the same (when an ngram language model is used), the lower scoring hypotheses can be dropped from the search. Hypothesis recombination makes the search more efficient by legitimately eliminating hypotheses (throughout the search) that cannot form part of the highest scoring translation according to the model (Koehn, 2009).

Heuristic Search

Since the number of possible translations for an input sentence is exponential in sentence length, an exhaustive scoring of all translations is not possible, and therefore a heuristic search method is used. Translation hypotheses are organised into stacks during the search so that when the number of hypotheses gets too large, similar hypotheses (stored in the same stack) can be compared with one another, and hypotheses that are less probable can be pruned from the search (Koehn, 2009). Each hypothesis stack contains translation hypotheses that were constructed by translating a specific number of foreign words. Two types of pruning commonly used are: histogram pruning and threshold pruning. For histogram pruning, a maximum number of hypotheses are kept in a stack, so that when this limit is exceeded lower scoring hypotheses are pruned away. Threshold pruning uses a fixed threshold, α, by which a translation hypothesis is allowed to be worse than the best hypothesis in the stack. If a translation hypothesis has a probability α times lower than the best translation it will be pruned away (Koehn, 2009).

Reordering Limit

In order to reduce the computational complexity of decoding when any reordering is allowed, when translating phrases out of sequence a reordering limit is imposed, where a maximum of d words may be skipped in the foreign sentence (Koehn, 2009).

Future Cost Estimation

Future cost estimation is used in decoding to try to minimise the chance of a lower scoring hypothesis being pruned because its score is not really comparable with the other hypotheses in its stack. For example, if $hypothesis_1$ was produced by translating the same number of foreign words as $hypothesis_2$, it's possible that it will have a lower probability simply because the foreign words $hypothesis_1$ translated are different from the foreign words translated in $hypothesis_2$. In order to level up the playing field a bit, for such cases, a future cost estimate is used that takes into account an estimate of the likely cost of translating the rest of the foreign input for each translation hypothesis (Koehn, 2009).

2.4 Existing Deep Syntax Transfer-based Machine Translation Systems

In this section, we critically review previous work that uses deep syntax as the intermediate representation for transfer in SMT (Riezler and Maxwell, 2006; Bojar and Čmejrek, 2007; Bojar and Hajič, 2008; Bojar, 2009). Riezler and Maxwell (2006) use the LFG f-structure as the intermediate representation in a transfer-based MT system that, like our own approach, applies standard methods of PB-SMT to deep syntactic transfer. Bojar and Hajič (2008), on the other hand, apply the Synchronous Tree Substitution Grammar (STSG) formalism of Hajič et al. (2002); Eisner (2003); Čmejrek (2006) to deep syntactic transfer MT and use the Tectogrammatical layer (T-layer) of Functional Generative Description (Sgall et al., 1986) (FGD), a labelled ordered dependency structure, as the intermediate representation for transfer. We continue with a description of each important component used by Riezler and Maxwell (2006) and Bojar and Hajič (2008), and provide motivation for our own approach. In addition, where relevant we include a comparison of the approach presented in Menezes and Richardson (2001).

2.4.1 Transfer Rule Extraction

Menezes and Richardson (2001) carry out transfer rule extraction from deep syntax parsed bilingual corpora using a combination of manually and automatically constructed bilingual dictionaries and an alignment grammar that establishes an alignment between nodes before extracting transfer rules. Nodes belonging to a specific part of speech, such as noun or verb are allowed to form boundaries of rules.

Riezler and Maxwell (2006), on the other hand, automatically align nodes using Giza++, using the surface form bitext corpus as input, symmetrizing with grow-diag-final and mapping the output onto the deep syntactic structures, before manually identifying systematic errors and automatically correcting them. They then extract an initial set of transfer rules constrained by the alignment between nodes and compute possible contiguous combinations of rules, with a maximum number of 3 primitive rules being used to form a new rule. In addition, a small number of hand-written transfer rules are used.

Bojar and Hajič (2008) automatically align nodes by firstly lemmatizing each side of the corpus and inputting this to Giza++ and use grow-diag-final for symmetrization. Transfer rules are extracted with an upper limit of at most 2 lexicalized nodes.

In our own approach, we carry out word alignment fully automatically and extract all rules consistent with the node alignment, in order to achieve high coverage of unseen data provided by including smaller rules, and to produce TL structures that contain fluent combinations of words, provided by larger rules that include more context. We also do not use any manually constructed transfer rules. The transfer decoder we use is trained fully automatically, as we are interested in learning how to translate for any language pair. We provide an critical review of the two rule extraction methods closest to our approach, Riezler and Maxwell (2006) and Bojar and Hajič (2008).

LFG F-structure Transfer Rules

Riezler and Maxwell (2006) automatically extract deep syntax transfer rules from a LFG parsed bilingual corpus and use these rules in the transfer step of a parse-transfer-generate pipeline. For rule extraction, they automatically word align the surface form bitext corpus in both language directions using Giza and symmetrize the word alignment using the *grow-diag-final* algorithm (described in Section 2.3.6). They then map the resulting word alignment onto the f-structures resulting in each

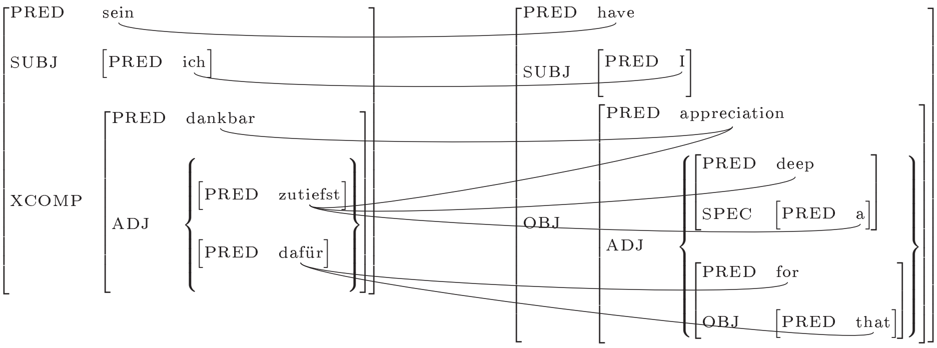


Figure 2.4: Word-aligned example f-structure pair taken from Riezler and Maxwell (2006)

local f-structure for both languages being aligned to between zero and many local f-structures. Figure 2.4 shows an example aligned f-structure pair for the German-English sentence pair: *"Ich bin zutiefst dankbar dafür." - "I have a deep appreciation for that."*. They then extract all possible rules that comply with Contiguity Constraints 1 and 2:

Contiguity Constraint 1. *Source and target f-structures are each connected.*

Contiguity Constraint 2. *F-structures in the transfer source can only be aligned with f-structures in the transfer target and vice-versa.*

Applying Contiguity Constraints 1 and 2 to the example f-structure pair shown in Figure 2.4 results in the set of primitive transfer rules shown in Figure 2.5. Transfer rules can contain variables, X_i, in either side used for mapping arguments in the SL to their correct position in the TL f-structure during translation. Riezler and Maxwell (2006) impose a limit of at most three primitive rules combining to form a complex rule to reduce the worst-case number of rules extracted from exponential to quadratic. Figure 2.6 shows a complex rule formed from two primitive rules of Figure 2.5.

The Contiguity Constraints defined in Riezler and Maxwell (2006) are a good starting point for transfer rule extraction, however, an analysis of the transfer rules it produces for different sentence pairs provides motivation for providing a new definition for rule extraction.

(a)

$$\begin{bmatrix}\text{PRED} & \text{sein}\\ \text{SUBJ} & X_0\\ \text{XCOMP} & X_1\end{bmatrix} \rightarrow \begin{bmatrix}\text{PRED} & \text{have}\\ \text{SUBJ} & X_0\\ \text{OBJ} & X_1\end{bmatrix}$$

(b)

$$\begin{bmatrix}\text{PRED} & \text{ich}\end{bmatrix} \rightarrow \begin{bmatrix}\text{PRED} & \text{I}\end{bmatrix}$$

(c)

$$\begin{bmatrix}\text{PRED} & \text{dankbar}\\ \text{ADJ} & \left\{\begin{matrix}\begin{bmatrix}\text{PRED} & \text{zutiefst}\end{bmatrix}\\ X_0\end{matrix}\right\}\end{bmatrix} \rightarrow \begin{bmatrix}\text{PRED} & \text{appreciation}\\ \text{SPEC} & \begin{bmatrix}\text{PRED} & \text{a}\end{bmatrix}\\ \text{ADJ} & \left\{\begin{matrix}\begin{bmatrix}\text{PRED} & \text{deep}\end{bmatrix}\\ X_0\end{matrix}\right\}\end{bmatrix}$$

(d)

$$\begin{bmatrix}\text{PRED} & \text{dafür}\end{bmatrix} \rightarrow \begin{bmatrix}\text{PRED} & \text{for}\\ \text{OBJ} & \begin{bmatrix}\text{PRED} & \text{that}\end{bmatrix}\end{bmatrix}$$

Figure 2.5: Primitive transfer rules produced by applying Contiguity Constraints 1 and 2 of Riezler and Maxwell (2006) to f-structure in Figure 2.4

$$\begin{bmatrix}\text{PRED} & \text{sein}\\ \text{SUBJ} & \begin{bmatrix}\text{PRED} & \text{ich}\end{bmatrix}\\ \text{XCOMP} & X_0\end{bmatrix} \rightarrow \begin{bmatrix}\text{PRED} & \text{have}\\ \text{SUBJ} & \begin{bmatrix}\text{PRED} & \text{I}\end{bmatrix}\\ \text{OBJ} & X_0\end{bmatrix}$$

Figure 2.6: Complex Transfer of Riezler and Maxwell (2006) constructed from primitive transfer rules in Figure 2.5(a) and 2.5(b)

The Contiguity Constraints are based on the definition of a consistent phrase in PB-SMT (described in Section 2.3.7). However, the Contiguity Constraints differ from the PB-SMT consistent phrase definition, in that they do not ensure that a transfer rule contains at least one aligned pair of nodes (in PB-SMT this corresponds to the part of Definition 1 of a consistent phrase that states that a consistent phrase includes at least one alignment point). This part of the definition of a consistent phrase in PB-SMT ensures a consistent phrase cannot be empty on either side and that unaligned words in source and target do not form phrases on their own. For example, Figure 2.7 shows the f-structures for the sentence pair *"Der Mitarbeiter des Monats hat Marie ja gern."* - *"The employee of the month likes Marie."* containing unaligned words *haben*, *ja* and *of*. Applying the Contiguity Constraints to this f-structure pair results in a set of primitive transfer rules which includes, for example, the two transfer rules shown in Figure 2.8, where the LHS of the transfer rule is empty, and Figure 2.9, where a pair of unaligned words in source and target form an erroneous transfer rule. Allowing transfer rules with an empty LHS is undesirable, because any such rule can be applied to *any* SL structure and would result in the possibility of adding *any* unaligned word of the target side of the corpus to *all* translations. Other deep syntax approaches that allow empty-sided transfer rules include Buch-Kromann (2007).

Another problem with the Contiguity Constraints is that they do not constrain the introduction of variables to transfer rules and subsequently allow transfer rules that contain singleton variables. A singleton variable is a variable that appears in one side of a transfer rule but not the other. For example, for the f-structure pair in Figure 2.7 the transfer rule in Figure 2.10 which contains the singleton variables X_0 and X_2 is allowed by the Contiguity Constraints. A transfer rule with a singleton variable in the LHS will produce a fragmented TL structure, in the RHS will produce a missing argument in the TL structure.

Although the Contiguity Constraints of Riezler and Maxwell (2006) are sufficient for capturing some types of translational divergence that can exist between the f-

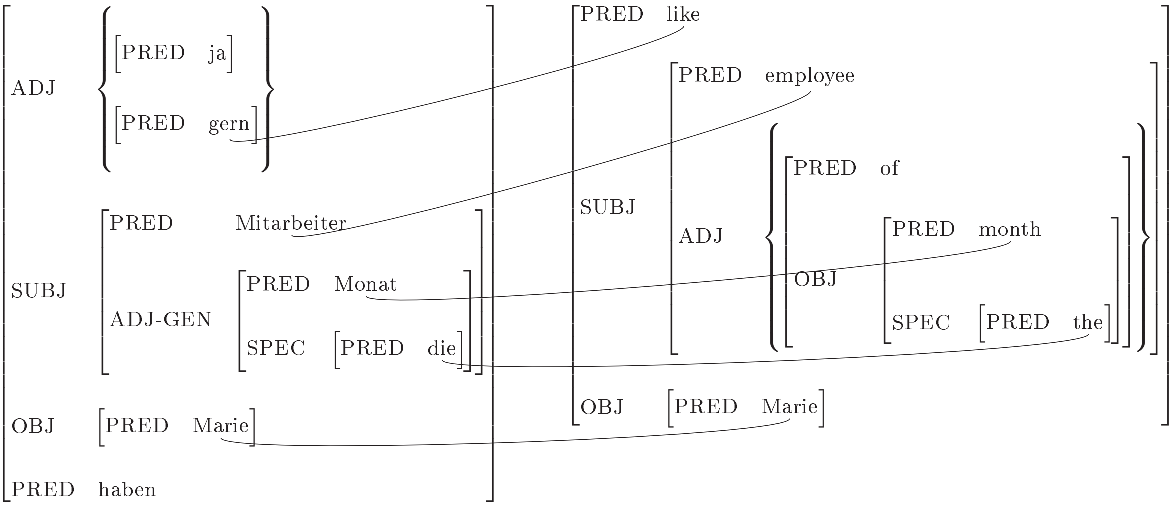


Figure 2.7: Example f-structure pair with unaligned local f-structures for the sentence pair *"Der Mitarbeiter des Monats hat Marie ja gern."*. and *"The employee of the month likes Marie."*

$$\{\} \quad \rightarrow \quad \begin{bmatrix}\text{PRED} & \text{of}\end{bmatrix}$$

Figure 2.8: Transfer rule with empty LHS extracted from f-structure pair of Figure 2.7 that complies with Contiguity Constraints 1 and 2 of Riezler and Maxwell (2006)

structures of a sentence pair, like in the argument switching example shown in Figure 2.4, when head-switching occurs across an f-structure pair, some erroneous transfer rules are allowed. For example, the f-structure pair shown in Figure 2.7 contains an example of head-switching: the local German f-structure with predicate *Marie* has *haben* as its head, whereas the corresponding local English f-structure *Marie* has *like* as its head, and *haben* ≠ *like*. From this example, the transfer rule shown in Figure 2.11 results. This transfer rule does not effectively transfer the SL local f-structure to the correct position in the TL structure.

This analysis motivates us to provide a new definition for consistent transfer rules, described later in Section 3.3.2.

FGD T-Layer Transfer Rules

Bojar and Hajič (2008) use the term *treelet pair* to describe transfer rules and define a treelet pair, $t_{1:2}$, as a tuple (t_1, t_2, m) where:

$$\begin{bmatrix}\text{PRED} & \text{ja}\end{bmatrix} \rightarrow \begin{bmatrix}\text{PRED} & \text{of}\end{bmatrix}$$

Figure 2.9: Example erroneous transfer rule extracted from f-structure pair of Figure 2.7 that complies with Contiguity Constraints 1 and 2 of Riezler and Maxwell (2006)

$$\begin{bmatrix}\text{PRED} & \text{haben} \\ \text{SUBJ} & \begin{bmatrix}\text{PRED} & \text{Mitarbeiter} \\ \text{ADJ-GEN} & X_0\end{bmatrix} \\ \text{OBJ} & \text{X}_1 \\ \text{ADJ} & \left\{\begin{matrix}\begin{bmatrix}\text{PRED} & \text{gern}\end{bmatrix} \\ \begin{bmatrix}\text{PRED} & \text{ja}\end{bmatrix}\end{matrix}\right\}\end{bmatrix} \rightarrow \begin{bmatrix}\text{PRED} & \text{like} \\ \text{SUBJ} & \begin{bmatrix}\text{PRED} & \text{employee} \\ \text{ADJ} & \left\{\text{X}_2\right\}\end{bmatrix} \\ \text{OBJ} & \text{X}_1\end{bmatrix}$$

Figure 2.10: Example transfer rule containing singleton variables extracted from f-structure pair of Figure 2.7 that complies with Contiguity Constraints 1 and 2 of Riezler and Maxwell (2006)

$$\begin{bmatrix}\text{PRED} & \text{gern}\end{bmatrix} \rightarrow \begin{bmatrix}\text{PRED} & \text{like} \\ \text{SUBJ} & \text{X}_0 \\ \text{OBJ} & \text{X}_1\end{bmatrix}$$

Figure 2.11: Example erroneous transfer rule extracted from f-structure pair of Figure 2.7 that complies with Contiguity Constraints 1 and 2 of Riezler and Maxwell (2006)

- t_1 and t_2 are source and target language treelets, respectively;
- m is a one-to-one mapping between frontier nodes in t_1 and t_2.

Given a set of states, Q, and a set of word labels, L, a treelet, t, is defined as a tuple (V, V^i, E, q, l, s) where

- $q \in Q$ is the root state of the treelet;
- V is a set of nodes;
- V^i is a non-empty set of internal nodes, such that $V^i \subseteq V$;
- V^f is a set of frontier nodes, such that $V^f = V \setminus V^i$;
- $E \subseteq V^i \times V$ is a set of directed edges forming a directed acyclic graph;
- l: $V^i \rightarrow L$, where L is a set of labels;
- s: $V^f \rightarrow Q$.

A synchronous derivation $\delta = \{t^0_{1:2}, t^1_{1:2}, ..., t^k_{1:2}\}$ constructs a pair of dependency trees, (T_1, T_2), by

- attaching treelet pairs $t^0_{1:2}$,...,$t^k_{1:2}$ at corresponding frontier nodes;
- ensuring that the root states of the attached treelet pairs matches the frontier states of the corresponding frontier nodes.

In order to directly compare the deep syntax transfer rules of Riezler and Maxwell (2006) and Bojar and Hajič (2008), consider the following example German to English translations of *Johannes mag Marie* (John likes Mary) and *Marie mag Johannes* (Mary likes John). Figures 2.12(a), 2.12(b) and 2.12(c) show Riezler and Maxwell (2006) transfer rules for translating the German words *mögen*, *Johannes* and *Marie* into English, Figures 2.12(d), 2.12(e) and 2.12(f) show STSG rules for translating the same German words and Figures 2.12(g) and 2.12(h) show the deep syntax structures for the German sentences *Johannes mag Marie* and *Marie mag*

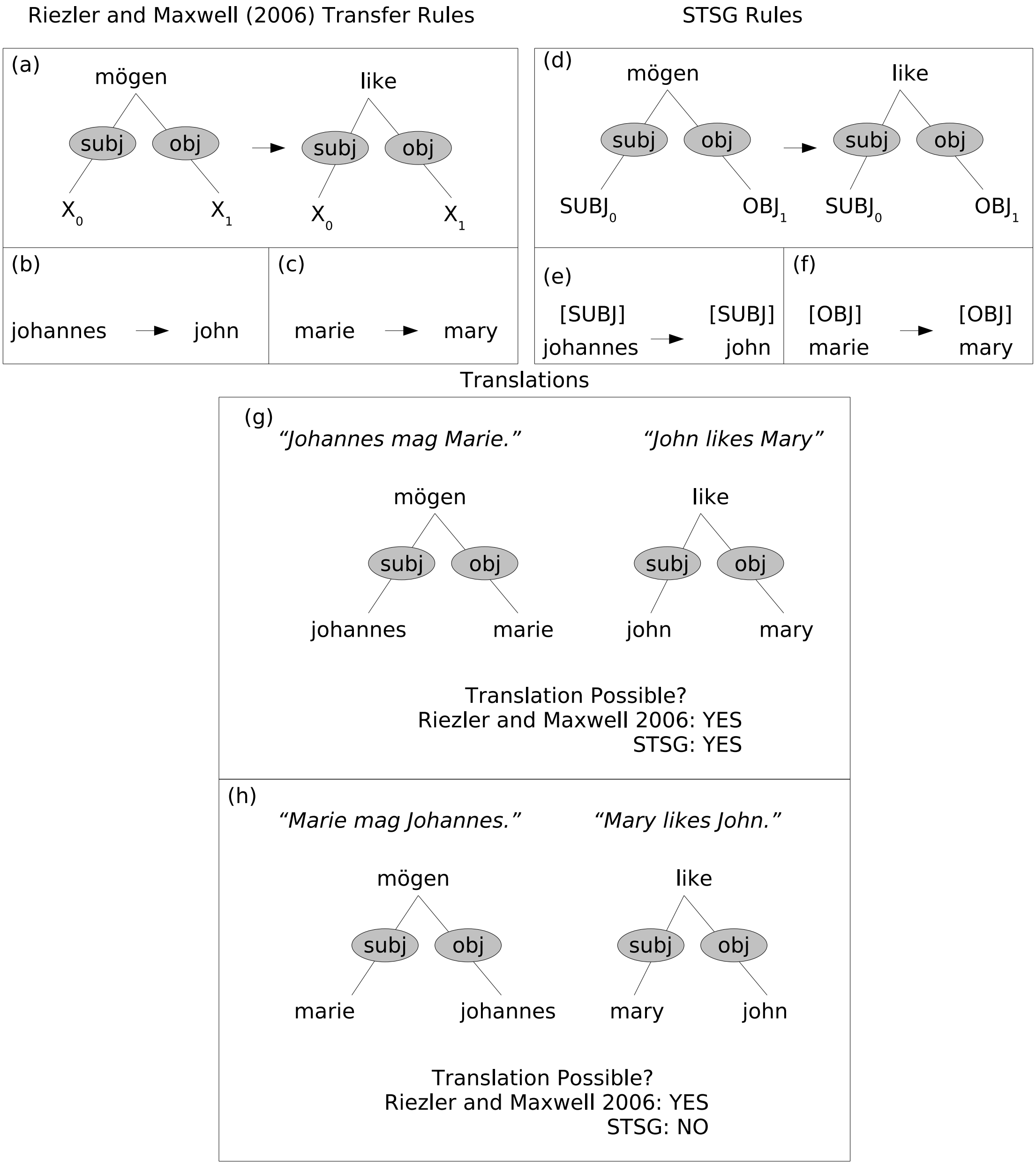


Figure 2.12: Riezler and Maxwell (2006) and STSG Deep Syntax Transfer Rules Comparison

Johannes and their translations. Each STSG rule has a single *start node* and 0-to-many *frontier nodes*. Both start nodes and frontier nodes each have a *state* and when two rules are used to translate two adjoining pieces of SL structure, the frontier state and start state where the rules join are required to unify with each other. For example, when the STSG rules of Figures 2.12(d), 2.12(e) and 2.12(f) are used to translate the German structure shown in Figure 2.12(g), for *Johannes mag Marie*, the respective frontier states of rule 2.12(d) (SUBJ and OBJ) must match the start states of the adjoining rules 2.12(e) (SUBJ) and 2.12(f) (OBJ). For example, since rule 2.12(d) has frontier node $SUBJ_0$ with state $SUBJ$, the start state of the rule 2.12(e) used to translate *Johannes* must also be $SUBJ$.

The constraint on frontier nodes and start nodes matching one another is not used in Riezler and Maxwell (2006). Constraining the use of rules via states, as is done in STSGs, means that the rules cannot be used in as many cases for unseen structures. For example, when translating the German structure shown in Figure 2.12(h) for *Marie mag Johannes* the STSG rules of Figures 2.12(d), 2.12(e) and 2.12(f) are not sufficient to translate the structure since the state labels of rules 2.12(f) (OBJ) and 2.12(e) (SUBJ) do not match the respective frontier states of rule 2.12(d) (SUBJ and OBJ), whereas the less constrained rules shown in Figures 2.12(a), 2.12(b) and 2.12(c) are sufficient for translating the same structure.[5]

In STSG, corresponding pairs of frontier nodes in the source and target side of a transfer rule are not required to have matching states and this enables the rules to capture certain translational divergence phenomena between deep syntax structures, such as argument-switching. For example, Figure 2.13(a) shows an example training pair that contains argument-switching. Figure 2.13(c) shows a STSG rule that correctly transfers the arguments by switching the subject and object when translating the verb *gefallen* to *like*. Figure 2.13(b) shows the equivalent rule of Riezler and Maxwell (2006) which can also transfer the divergent structure correctly.

Bojar and Hajič (2008) only allow rules that have a one-to-one mapping between

[5] Bojar and Hajič (2008) investigate effects of relaxing this STSG constraint.

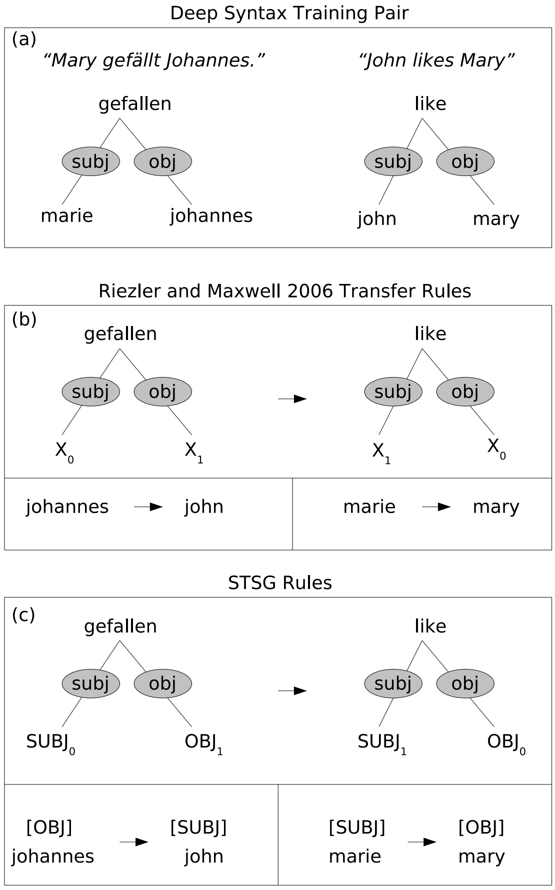


Figure 2.13: Example rules of Riezler and Maxwell (2006) and STSG both capturing argument-switching

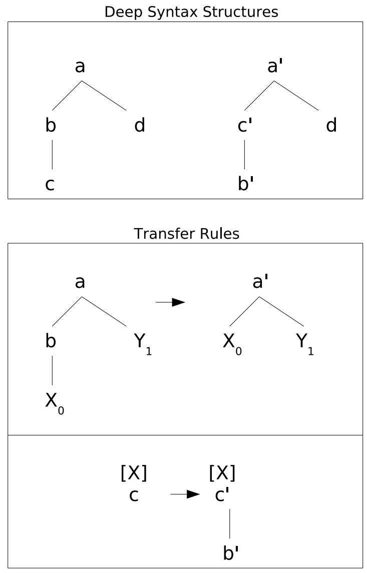


Figure 2.14: Erroneous STSG rules caused by head-switching

frontier nodes, thus eliminating rules that contain singleton variables, unlike Riezler and Maxwell (2006). In addition, in contrast to Riezler and Maxwell (2006), Bojar and Hajič (2008) add the constraint that a rule must contain at least one internal node, thus eliminating rules with one side empty.

Bojar and Hajič (2008) state that their method of extracting STSG rules should only be applied to structure pairs that have a certain degree of isomorphism. For example, Figure 2.14(a) shows two possible structures where translational equivalent nodes are labelled a and a', in which head-switching takes place across the pair of structures for node b, b', c and c'. The erroneous transfer rules shown in Figures 2.14(b) and 2.14(c) would result according to the definition in Bojar and Hajič (2008), if such a structure is found in training.

This analysis of previous work for automatic extraction of deep syntax transfer rules motivates a new definition for consistent transfer rules we provide later in Section 3.3.2. Our definition can be applied to both isomorphic and non-isomorphic deep syntax structure pairs without producing erroneous rules and disallows empty-sided rules. In addition, our method follows the approach of Riezler and Maxwell (2006) and does not constrain the use of rules during translation via frontier states.

2.4.2 Translation Models

Both Riezler and Maxwell (2006) and Bojar and Hajič (2008) use a log-linear combination of feature functions as their translation model. The features used by both Riezler and Maxwell (2006) and Bojar and Hajič (2008) are as follows:

- deep syntax translation model (both language directions);
- deep syntax language model;
- deep syntax phrase penalty: the number of transfer rules used to construct the target structure;
- deep syntax word penalty: the number of nodes in the target structure;
- surface form language model.

Additional features used in Riezler and Maxwell (2006) are:

- deep syntax lexical weighting (both language directions);
- the number of transfer rules with frequency 1;
- the number of default transfer rules;[6]
- number of constituent movements during generation based on the original order of the head predicates of the constituents;
- number of generation repairs;
- number of words in the generated string;

Additional feature used in Bojar and Hajič (2008) is:

- STSG synchronous derivation probability model.

The following is a list of the features we adopt from Riezler and Maxwell (2006) and Bojar and Hajič (2008) in our own approach and the motivation for doing so:

[6] A default transfer rule transfers a SL word as itself.

- deep syntax translation model (both language directions): this is essentially the translation model of PB-SMT applied to deep syntax;
- deep syntax language model: to help produce fluent combinations of words in TL structures;
- deep syntax phrase penalty: used for reasons similar to the motivation for using a phrase penalty in PB-SMT, so that larger rules can be preferred which helps produce fluent combinations of words in TL structures;
- deep syntax word penalty: to counter-balance the bias of the deep syntax language model toward smaller output structures;
- deep syntax lexical weighting: as a back-off to the translation model as it provides a richer model and better statistical estimates;
- surface form language model: to model fluency in the surface form string after generation;

Besides these, we also use some additional features and these are discussed in detail in Chapter 4.

Deep Syntax Language Models

Some of the features used in Bojar and Hajič (2008) and Riezler and Maxwell (2006) are applied during decoding (when the TL deep syntactic structure is constructed) and some later after generation of the TL string. The features used during decoding have most influence over the final output. The decoding features influence what part of the immense search space is reached by the heuristic search and subsequently they determine the content of the n-best list of TL structures. If an important feature is not used during decoding, like the language model for example, this is likely to result in early elimination of good TL output structures. Bojar and Hajič (2008) use a deep syntax language model during decoding, while Riezler and Maxwell (2006) only apply their deep syntax language model after decoding on the n-best target

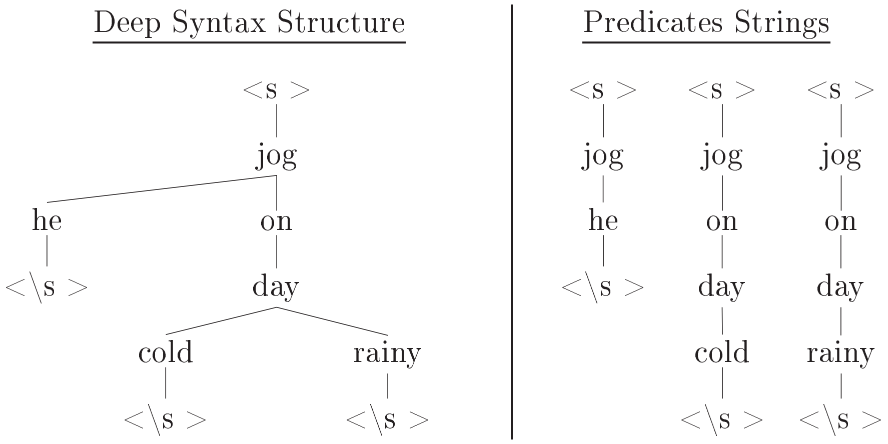


Figure 2.15: Example deep syntax structure for "He jogs on cold rainy days." where strings of predicates from root to frontier causes incorrect ngram counts

language structures. The approach of Bojar and Hajič (2008), using a language modeling during decoding, is preferable as this feature helps guide what part of the search space is reached and influences the quality of the n-best TL structures. If fluency is not taken into account during the search it is left to chance that the n-best list of TL structures will contain *any* structures that are fluent combinations of words.

In addition, the method presented in Riezler and Maxwell (2006) for computing the deep syntax language model probability will produce an incorrect result for certain structures. Riezler and Maxwell (2006) use *"the log probability of strings of predicates from root to frontier of target f-structure, estimated from predicate trigrams in English f-structures"* to compute the deep syntax language model probability for TL structures. For structures containing some unary branching followed by branching of arity greater than one, the language model probability will be incorrect. Figure 2.15 shows the deep syntax structure for the English sentence *"He jogs on cold rainy days."* that contains unary branching for the node sequence *jog-on-day*. If the language model probability is computed by simply combining the probability of the individual strings of predicates from root to frontier of the deep syntax structure, then, for example, $p(day|jog\ on)$ will be included twice and the overall probability estimate will be incorrect.

For deep syntax language modeling, our own approach follows the approach of Bojar and Hajič (2008) and uses a deep syntax language model *during decoding*, but we include more context in our model by using a trigram deep syntax language model as opposed to the bigram model of Bojar and Hajič (2008). In addition, when computing the deep syntax language model during training and when estimating the probability of TL structures during decoding, we ensure that no ngrams are duplicated, as in Riezler and Maxwell (2006), to avoid miscalculating deep syntax language model probability estimates for some TL structures. We further describe how we carry out deep syntax language modeling in Section 4.5.

2.5 Other Syntax-Based SMT Approaches

In our discussion, we focus on related work that uses deep syntax in SMT. A great deal of work has been carried out using other kinds of syntactic formalisms for MT also. For example, Galley et al. (2004) use a theory that gives formal semantics to word level alignments to introduce an algorithm that derives the minimal set of syntactically motivated transformation rules that explain human translation data, while Chiang (2007a) defines a hierarchical phrase-based machine translation model that achieves sophisticated reordering by allowing sub-phrases contained within larger phrases to be replace by a non-terminal in the source and target language. Zollmann and Venugopal (2006), on the other hand, use chart parse decoding that operates on phrase tables augmented with TL syntactic categories. They parse the TL side of the bilingual corpus with a phrase structure grammar and align them with phrase table lattices for corresponding source sentences. They use techniques to augment and generalize the phrase table by aligning SL phrases with TL syntactic categories to produce a synchronous bilingual grammar. Costs are assigned to the synchronous context free grammar using a log-linear model with weights optimized via MERT, and the following features: lexical weights, relative frequencies of rules, number of rule applications, number of TL words, rule type flags as well as a rareness and unbalancedness penalty. Liu et al. (2006) use a syntax-based Tree-to-string Alignment Template (TAT) model that is automatically extracted from aligned parallel corpora that has been parsed on the source side, while Marcu et al. (2006) define a machine translation model that uses syntactified target language phrases.

2.6 Conclusion

This chapter introduced the relevant theories behind our approach, such as LFG and PB-SMT, and also provided a discussion of previous research into combining deep syntax and SMT techniques. The chapter provided a detailed analysis of Riezler and Maxwell (2006) and Bojar and Hajič (2008) comparing how the two pieces of work use similar or different techniques and what parts we adopt in our own work and our motivation for doing so. The chapters that follow describe in detail how we combine techniques from PB-SMT with deep syntax.

Chapter 3

Deep Syntax Transfer Rules

3.1 Introduction

In this chapter, we describe a method of automatically extracting transfer rules from deep syntax parsed bitext corpora. A deep syntax transfer rule translates a snippet of SL deep syntactic structure into a corresponding TL structure snippet. Our approach aims to extract appropriate transfer rules from both isomorphic and non-isomorphic structures. We provide a definition of consistency for extracting deep syntax transfer rules that achieves this in a similar way to the method used to extract phrases in PB-SMT, by firstly establishing an alignment between nodes before extracting all rules consistent with this alignment. Since our definition of consistency allows up to an exponential number of rules to be extracted per training pair we also provide efficient methods of extracting and storing large numbers of rules.

3.2 Deep Syntax Transfer Rules

As in Riezler and Maxwell (2006), the transfer rules in our own work are composed of a LHS and RHS snippet of deep syntactic structure.[1] Each side of a transfer rule is made up of at least one lexicalized node and zero or more non-lexicalized nodes. Lexicalized nodes are labeled with their predicate value and a set of atomic feature-value pairs, whereas non-lexicalized nodes are indexed variables, X_i, used for transferring the arguments of a SL lexicalized node to the correct position in the TL structure. The dependency relations between the nodes of the deep syntactic structure are present as labels on the arcs of the dependency graph. Figure 3.1 shows an example German to English transfer rule, extracted from deep syntactic structures for the sentence pair *"Er kommt gut voran."* and *"He is progressing nicely."*, in which the LHS contains a single lexicalized node with predicate value *(voran)kommen* with two arguments, a *subject* (X_0) and an *adjunct* (X_1) and the

[1] Since we use the underlying graph structure of deep syntactic structures to hypothesize transfer rules, we illustrate each side of a rule as a graph as opposed to AVMs, as in Riezler and Maxwell (2006).

RHS consists of a single lexicalized node with predicate value *progress* and a *subject* (X_0) and *object* (X_1) argument. This rule transfers between isomorphic pieces of SL and TL structure, since the labeled graph structure and variables are identical in both sides of the rule. Figure 3.1(b), on the other hand, captures a non-isomorphic correspondence, extracted from the deep syntactic structures of the sentence pair *"Wir halten das für gut."* and *"That is good."*, in which the translation involves a degree of paraphrasing. The LHS of the rule has German predicate *halten* as its root with three arguments, a *subject*, the lexicalized node with predicate *pro* (representing a pronoun), an *object* (X_0) and an *xcomplement-predicate*, the lexicalized node with predicate *für* and *object* X_1. The RHS of the rule has *be* as its root with two arguments, a *subject* (X_0) and an *xcomplement* (X_1).

3.3 Transfer Rule Extraction

The first step in transfer rule extraction is to automatically align the nodes of each pair of deep syntactic structures in the training corpus. Any method of automatic node alignment can be used. Here, we provide motivation for and describe one method of automatically aligning the nodes of deep syntactic structures.

3.3.1 Automatic Node-Level Alignment of Deep Syntactic Structures

Since our main interest is to build a system that automatically learns how to translate from one language to another for any language pair, we carry out node-level alignment fully automatically (Bojar, 2009), unlike other approaches that use bilingual dictionaries for translating lexical items (Carl, 2007), or approaches that carry out automatic word alignment, before manually detecting systematic errors and automatically correcting them (Riezler and Maxwell, 2006).

Figure 3.2 shows how Riezler and Maxwell (2006) carry out automatic alignment of the nodes of the deep syntactic structures in the bitext training corpus by firstly

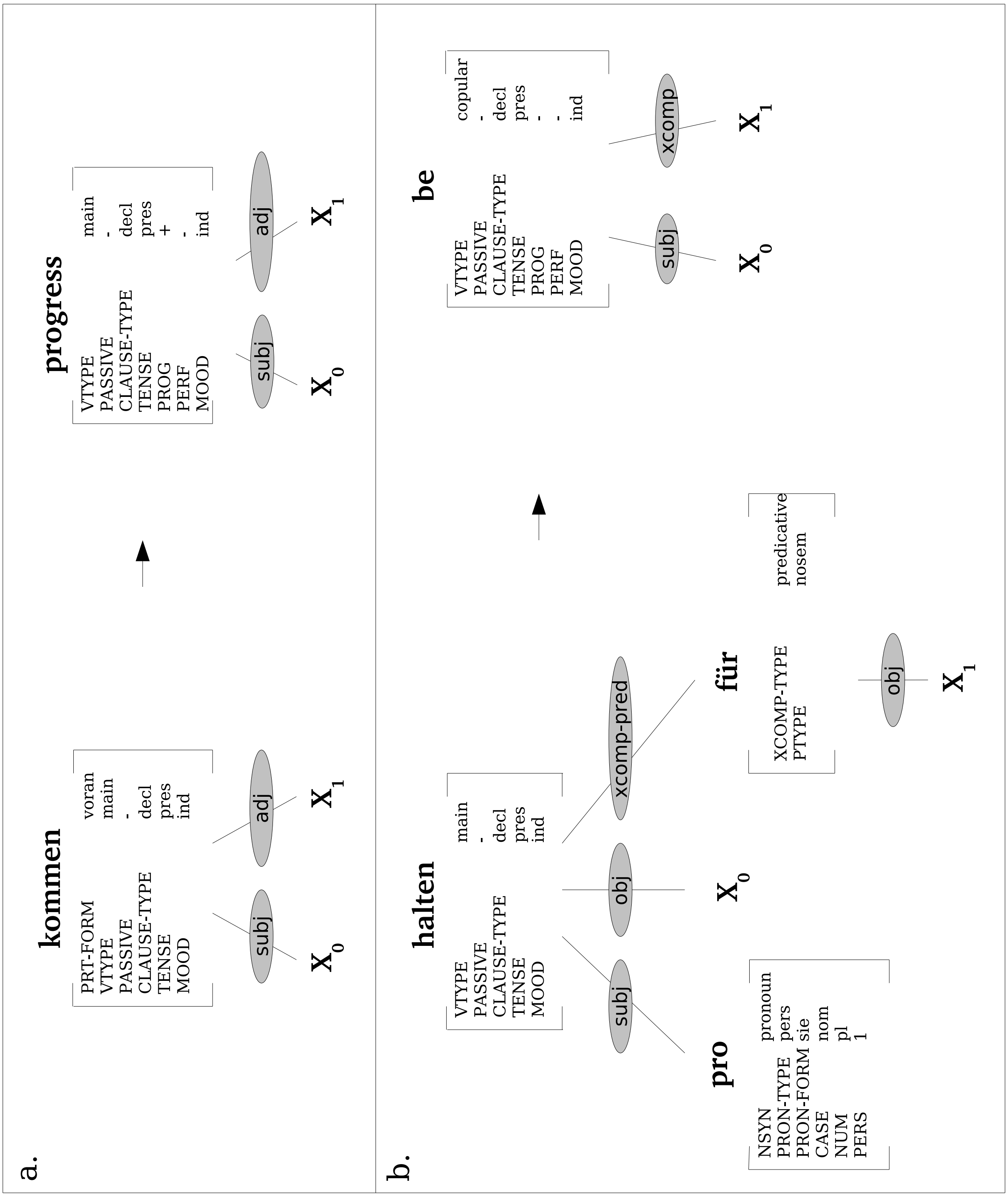


Figure 3.1: Example transfer rules: extracted from sentence pairs (a) *"Er kommt gut voran."* and *"He is progressing nicely."* and (b) *"Wir halten das für gut."* and *"That is good."*

running Giza++ (Och et al., 1999) on the surface form bitext corpus to produce an alignment between surface form words, then parsing each side of the corpus before applying the surface form word alignment to the nodes of the deep syntactic structures.[2] Carrying out word alignment on the surface form bitext corpus, however, does not take full advantage of the more language independent representation of the training data, i.e. the deep syntactic structures. In addition, previous work on German to English word alignment has shown that word alignment performance is improved when inflectional morphology is normalized (Corston-Oliver and Gammon, 2004). Instead of carrying out word alignment on the surface form corpus, in our own approach, we run automatic word alignment on a version of the bitext corpus that is reconstructed from the deep syntax parsed training corpus.

Figure 3.3 shows a step-by-step illustration of how we reconstruct the bitext. The original surface form bitext, shown in Figure 3.3(a), is firstly parsed, Figure 3.3(b), then the predicate value of each node in the structure pair is extracted, Figure 3.3(c), with predicates ordered in each sentence according to a depth-first traversal of the deep syntactic structure.[3] For example, the order of the predicates in the reconstructed corpus of the German structure in Figure 3.3(b) is *verlieren hersteller kannen öl die schlüssel ihre.* The reconstructed bitext is then input to Giza++ (Och et al., 1999) and automatic word alignment is run in both language directions. The output is then input to Moses (Koehn et al., 2007) to compute the symmetrization of the bidirectional alignment, Figure 3.3(d). For example, the intersection of the bidirectional word alignment can be used, as it yields a reliable one-to-one alignment between nodes. Finally, the alignment is applied to the deep

[2] Note that in our examples from LFG containing the definite article, for example the definite article belonging to *Hersteller* in Figure 3.2(c), all instances of the definite article are represented by the predicate *die*.

[3] In order for the depth-first traversal not to loop if a structure contains instances of reentrancy or argument sharing we temporarily ignore these dependencies when reconstructing the corpus from the structures. In addition, although local f-structures are unordered, in the Prolog-encoded f-structures produced by the parser local f-structures are ordered in a systematic way, for example, for transitive verbs, the constraint that specifies that a word has a subject will appear before the constraint for its object. We take advantage of this and use the order of the arguments produced by the parser when doing the depth-first traversal for word-alignment.

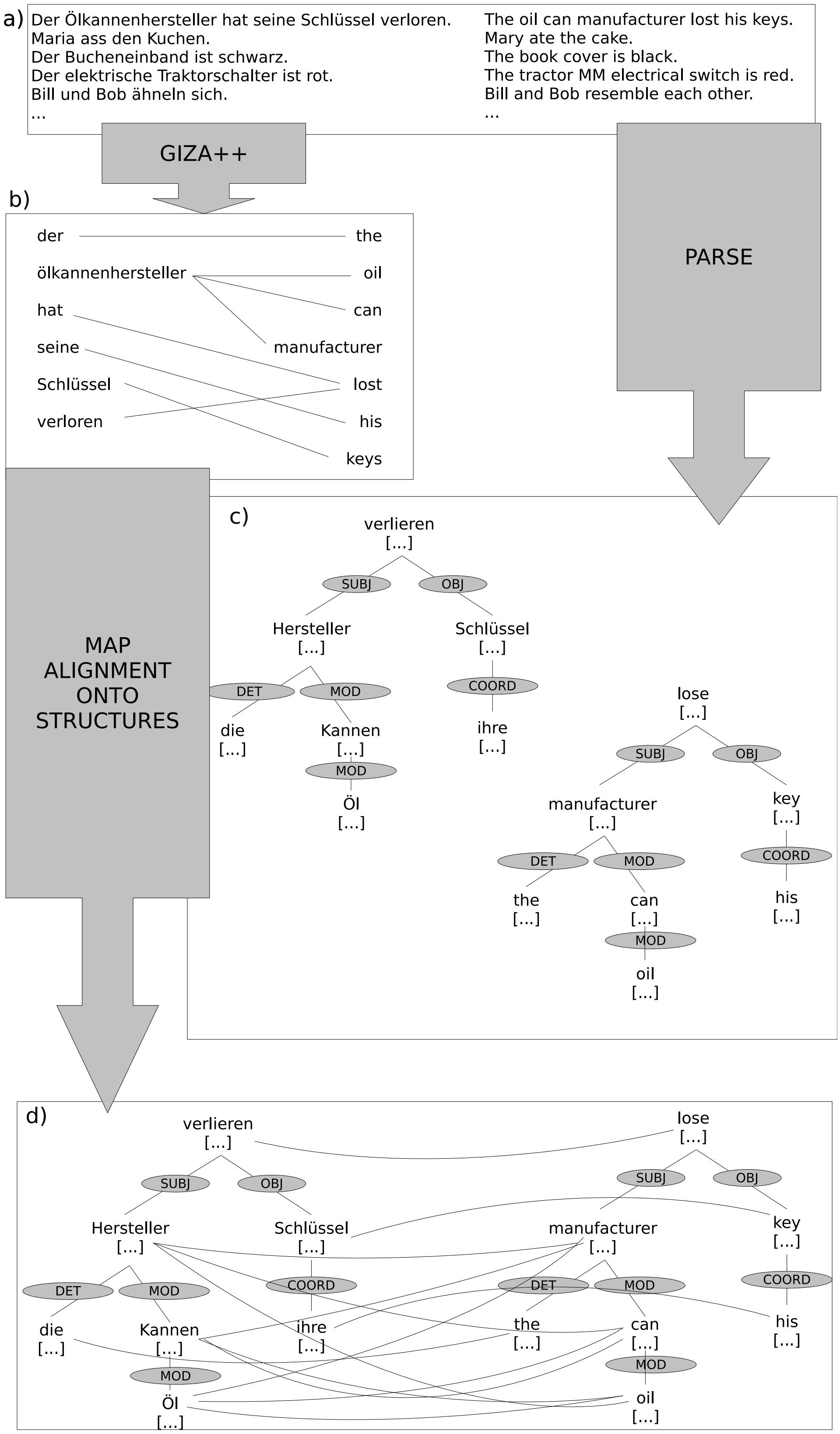


Figure 3.2: Automatic Alignment of Deep Syntactic Structures Nodes in Riezler and Maxwell (2006)

syntactic structures, Figure 3.3(e).

The advantages of carrying out alignment on the deep syntax corpus as opposed to the surface form corpus are as follows; (i) auxiliary verbs are not present in the deep syntax corpus and since we do not need to align them, including them in the input to automatic alignment unnecessarily increases the number of words that must be automatically aligned, (ii) the words in the deep syntax bitext are in lemma form and since this is a more general representation than the surface form it should help with data-sparseness problems, (iii) German compound nouns are resolved to their component nouns in the deep syntax corpus, which should correspond better to English words. In Chapter 6 we carry out an experimental comparison between these two methods of alignment.

3.3.2 Consistent Transfer Rules

As in Phrase-Based SMT, where a word alignment for each example sentence pair is first established before all phrases consistent with the word alignment are extracted (Och et al., 1999; Koehn et al., 2003), we extract all transfer rules that are consistent with the node alignment. For each deep syntax training pair, the rule extraction algorithm uses the underlying graph structure, as well as an alignment between nodes of the structure pair, to extract all transfer rules consistent with the node alignment.

We define a consistent transfer rule using a simplification of the actual training deep syntactic structures and temporarily consider them as tree structures by ignoring (i) arcs that cause cycles in the graph and (ii) arcs that share an end node with another arc. For example, if the subject of node A is also the subject of node B, one of these arcs is ignored temporarily. This is done by labeling the nodes using an increasing index in depth first order.[4] Then arcs with an end node label less than their start node are ignored.

Consistent Rule Definition 1. *Given an alignment, A, between nodes in depen-*

[4] Labeling is carried out by the parser.

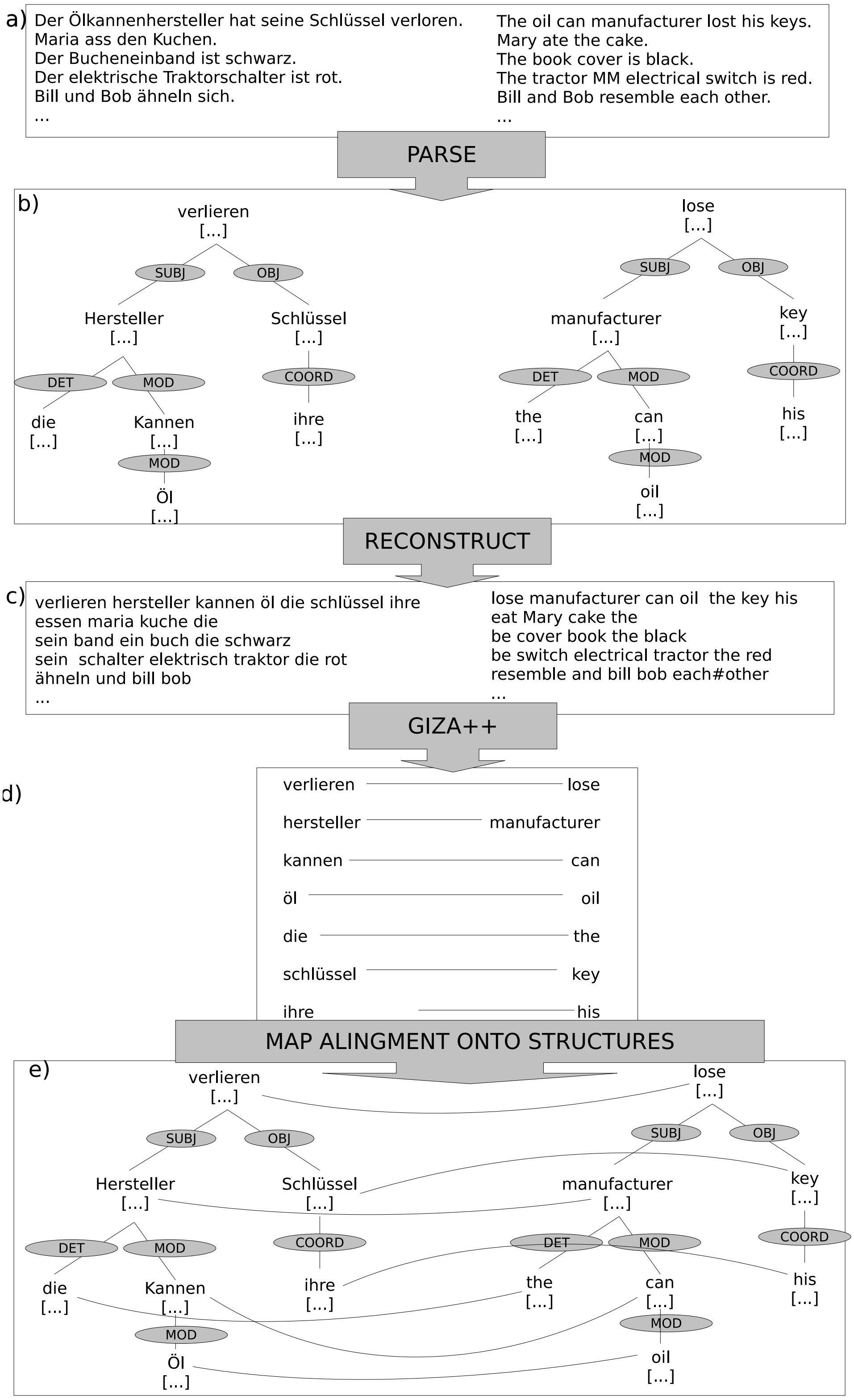

a)
Der Ölkannenhersteller hat seine Schlüssel verloren.
Maria ass den Kuchen.
Der Bucheneinband ist schwarz.
Der elektrische Traktorschalter ist rot.
Bill und Bob ähneln sich.
...
The oil can manufacturer lost his keys.
Mary ate the cake.
The book cover is black.
The tractor MM electrical switch is red.
Bill and Bob resemble each other.
...
PARSE
b)
verlieren [...]
SUBJ
OBJ
Hersteller [...]
Schlüssel [...]
DET
MOD
COORD
die [...]
Kannen [...]
ihre [...]
MOD
Öl [...]
lose [...]
manufacturer [...]
key [...]
the [...]
can [...]
his [...]
oil [...]
RECONSTRUCT
c)
verlieren hersteller kannen öl die schlüssel ihre
essen maria kuche die
sein band ein buch die schwarz
sein schalter elektrisch traktor die rot
ähneln und bill bob
...
lose manufacturer can oil the key his
eat Mary cake the
be cover book the black
be switch electrical tractor the red
resemble and bill bob each#other
...
GIZA++
d)
verlieren — lose
hersteller — manufacturer
kannen — can
öl — oil
die — the
schlüssel — key
ihre — his
MAP ALINGMENT ONTO STRUCTURES
e)

Figure 3.3: Automatic Alignment of Deep Syntactic Structures Nodes

dency pair (F, E), $(\overline{f}, \overline{e})$ is a rule consisting of nodes (N_f, N_e), rooted at (r_f, r_e), with descendents (D_f, D_e) of r_f and r_e in F and E respectively, if

$$\begin{aligned}
& N_f = r_f \cup D_f \\
\bigwedge\ & N_e = r_e \cup D_e \\
\bigwedge\ & \forall f_i \in N_f : (f_i, e_j) \in A \rightarrow e_j \in N_e \\
\bigwedge\ & \forall e_j \in N_e : (f_i, e_j) \in A \rightarrow f_i \in N_f \\
\bigwedge\ & \exists e_j \in N_e : (r_f, e_j) \in A \\
\bigwedge\ & \exists f_i \in N_f : (f_i, r_e) \in A
\end{aligned}$$

Consistent Rule Definition 2. *For any rule $(\overline{f}, \overline{e})$ in dependency pair (F, E) rooted at (r_f, r_e) consisting of nodes N_f and N_e, where $(\overline{s}, \overline{t})$ is also a rule in (F, E) rooted at (r_s, r_t) consisting of nodes N_s and N_t where $r_s \neq r_f$, $r_t \neq r_e$, if $r_s \in N_f$ and $r_t \in N_e$ then there is a rule $(\overline{a}, \overline{b})$ rooted at (r_f, r_e) with nodes r_s and r_t replaced by variable x_k, where k is an index unique to the transfer rule, consisting of nodes:*

$$\begin{aligned}
N_a &: (N_f \backslash N_s) \cup x_k \\
N_b &: (N_e \backslash N_t) \cup x_k
\end{aligned}$$

Definition 1 applied to a deep syntactic structure pair produces a set of initial rules containing no variables by identifying pairs of spanning subtrees within the pair of structures that correspond to one another according to the alignment between nodes. Figure 3.4 shows a node-aligned pair of deep syntactic structures for the sentence pair *"Der Herr des Hauses hat die Mehrheit bei der Abstimmung bekommen"* - *"The boss of the house received a majority vote"*, with aligned nodes labeled by the same index number. [5] Figure 3.5 shows all pairs of spanning subtrees for the structures that form rules according to Definition 1 and Figure 3.6 shows the initial set of transfer rules identified by applying Definition 1 to the deep syntactic structure pair of Figure 3.4. For example, the pair of spanning subtrees

[5] In the example, aligned nodes of the deep syntactic structure pair are in a one-to-one alignment but many-to-one and one-to-many alignment is also possible

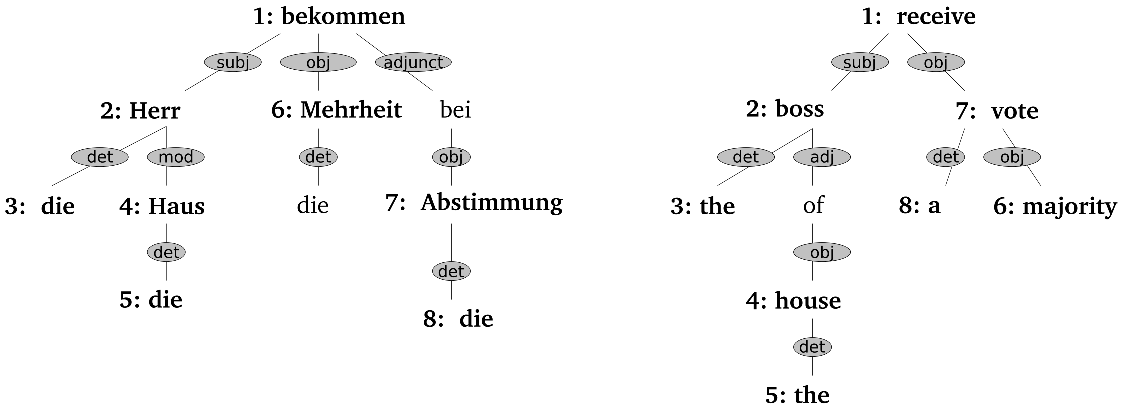


Figure 3.4: Example node-aligned deep syntactic structure pair for sentences *"Der Herr des Hauses hat die Mehrheit bei der Abstimmung bekommen"* - *"The boss of the house received a majority vote"* with pairs of aligned nodes labeled by matching id numbers.

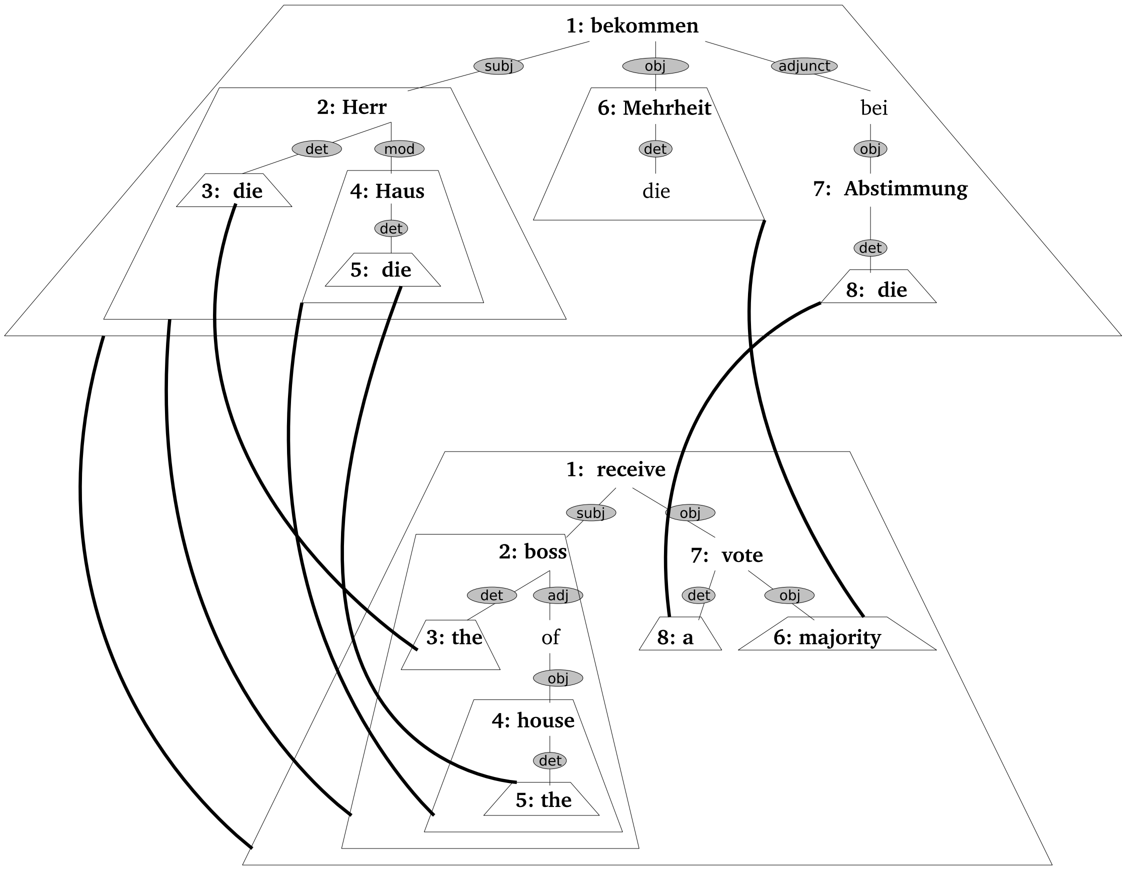


Figure 3.5: All spanning subtrees identified by Definition 1, illustrated as linked trapezoids, for deep syntactic structure pair of Figure 3.4

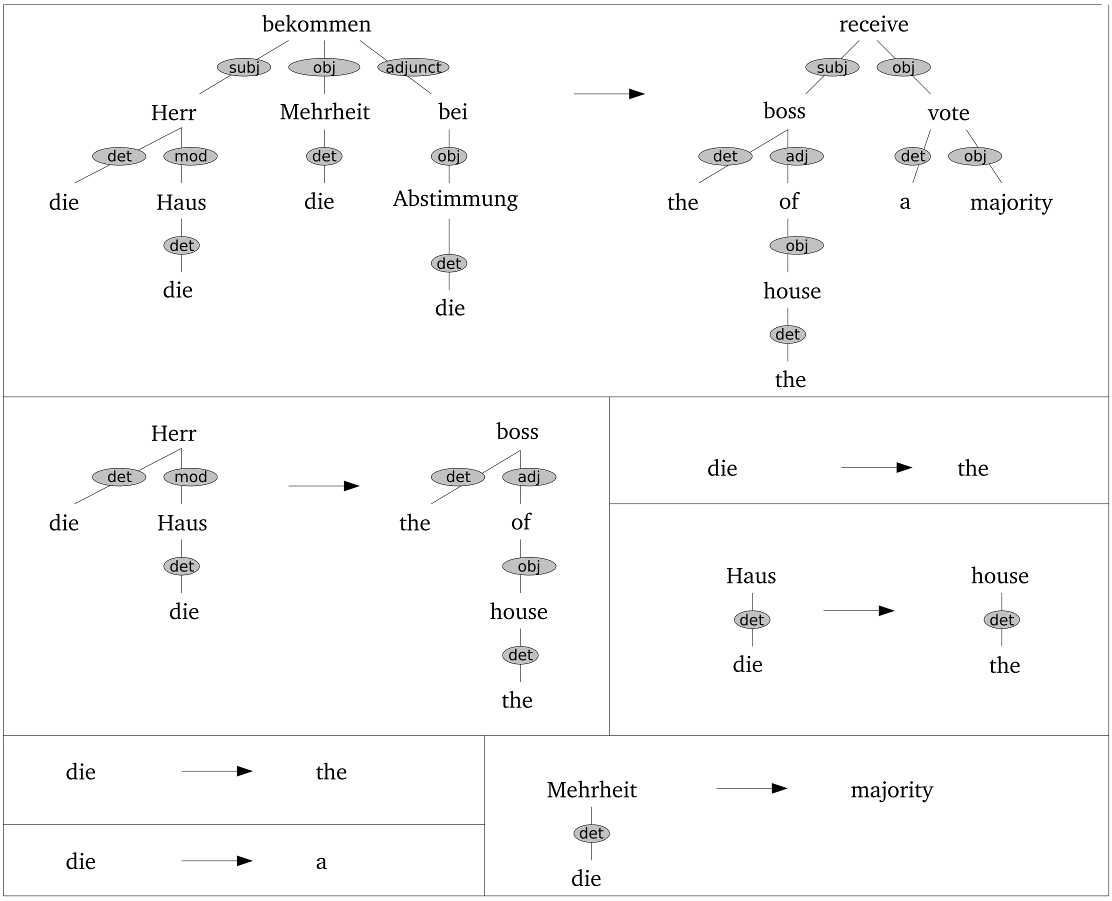


Figure 3.6: Initial set of transfer rules identified by Definition 1 for deep syntactic structure pair in Figure 3.4

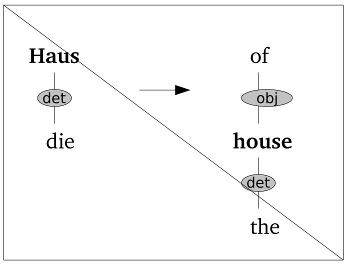


Figure 3.7: Potential transfer rule from deep syntactic structures in Figure 3.4 eliminated by constraint of Definition 1 in which root nodes in each side of transfer rules are each aligned nodes

containing nodes *Mehrheit* and *die* in German and *majority* in English form a rule according to Definition 1, where both spanning subtrees contain the same alignment points. There is no rule produced by Definition 1 rooted at nodes *Abstimmung* and *vote* because the spanning subtrees rooted at this pair of nodes do not contain the same set of alignment points, i.e. the following part of Definition 1 is not true for the spanning subtree due to the alignment between nodes *Mehrheit* and *majority* violating $\forall e_j \in N_e : (f_i, e_j) \in A \rightarrow f_i \in N_f$.

The constraints included in Definition 1, enforcing each root node of the spanning subtrees that form a rule to each be an *aligned* node, disallow an unaligned node to be the root node of an initial rule and subsequently cause each unaligned node in the training pair to remain always adjoined to its head. Figure 3.7 shows a potential transfer rule that contains the same set of aligned nodes but is ruled out by this constraint, due to the root node of the English structure *of* being an unaligned node. Since *of* is unaligned, it will only be part of a rule also containing its head in the English structure *boss*. This constraint also eliminates rules with one empty side and erroneous rules that transfer any unaligned node in the source structure to any unaligned node in the target structure, which is undesirable for reasons we described in Section 2.4.1.

The constraint on root nodes of transfer rules each being an aligned node does *not* mean that pairs of root nodes must align with *each other*, however. This al-

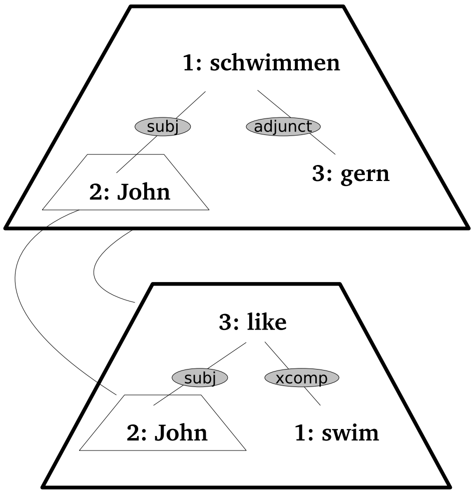


Figure 3.8: Deep syntactic structure pair for sentences *"John schwimmt gern."* and *"John likes to swim."* where head-switching occurs between *schwimmen* and *gern* in German and *like* and *swim* in English, aligned nodes are labeled with the same id number

lows Definition 1 to still yield appropriate rules when head-switching occurs within a pair of structures. For example, Figure 3.8 shows the deep syntactic structures for the sentence pair *"John schwimmt gern."* and *"John likes to swim."* in which head-switching occurs between *schwimmen* and *gern* in German and *like* and *swim* in English. The spanning subtrees rooted at *schwimmen* and *like* form a rule by Definition 1 since the spanning subtrees rooted at each node contain the same alignment points and *schwimmen* and *like* are each aligned nodes (even though they are not aligned to each other).

Definition 2 yields additional transfer rules by allowing a rule that is nested within a larger rule to be replaced by a single variable, X_i, in the LHS and RHS. Since the initial set of rules produced by Definition 1 are corresponding spanning subtrees, we can legitimately replace any pair of subtrees with a variable. As we showed in Section 2.4.1, the alternate method of introducing variables to transfer rules used in Riezler and Maxwell (2006) and Bojar and Hajič (2008) that allows any pair of aligned nodes to be replaced by a variable produces erroneous transfer rules when head-switching or other kinds of non-isomorphism exists between a pair

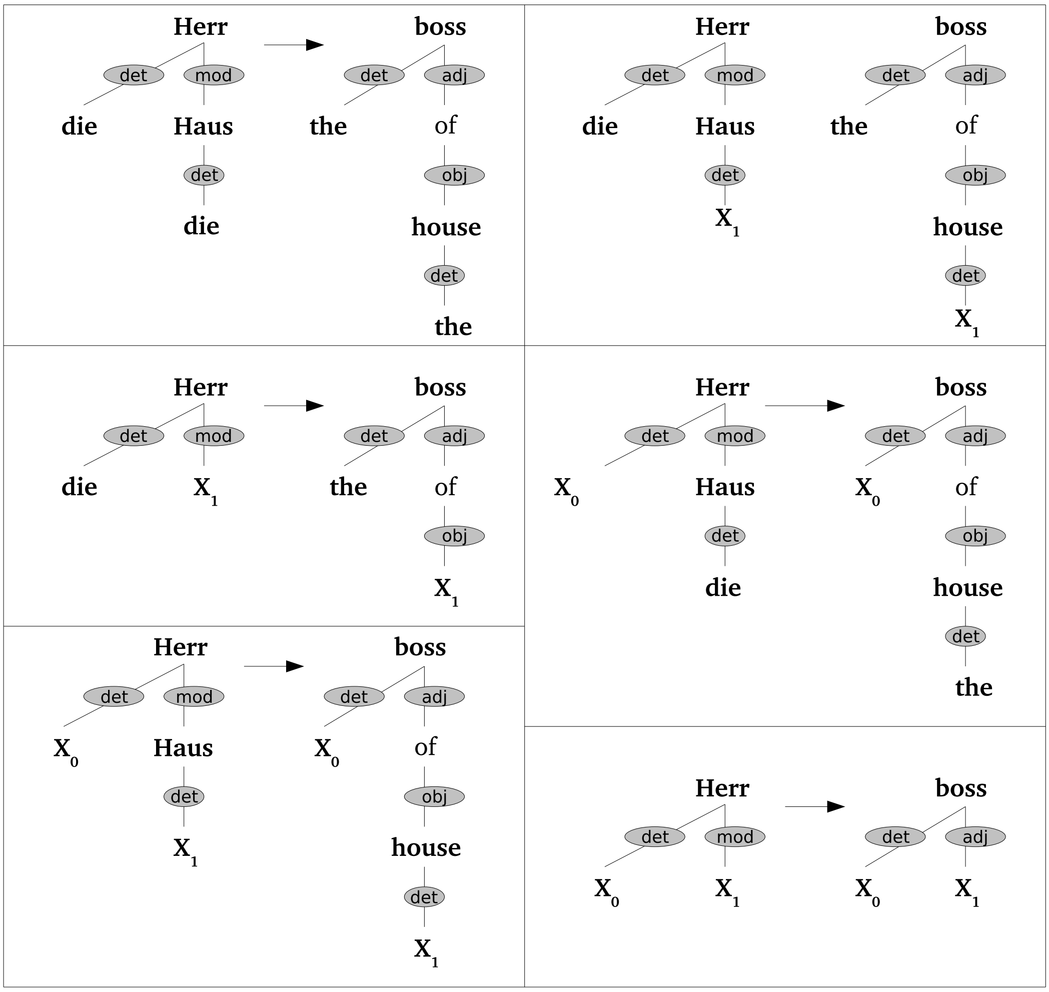


Figure 3.9: All transfer rules rooted at node pair *Herr* and *boss* for structures of Figure 3.4

of structures. Figure 3.9 shows all transfer rules for the deep syntactic structures of Figure 3.4 rooted at the nodes *Herr* and *boss* produced by Definition 1 and Definition 2.

Complexity

Our motivation for extracting *all* rules consistent with the node alignment is similar to that of PB-SMT, where all phrases consistent with the word alignment are extracted: smaller rules help to achieve high coverage of unseen data (i.e. unseen SL deep syntactic structures) and larger rules provide fluent combinations of TL words. The number of transfer rules produced by our definition for a rule consistent with the node alignment is constrained by both the number of aligned nodes in the deep syntactic structure pair and the level of isomorphism between the two structures. In general, the more isomorphic the pair of structures and the more nodes aligned the higher the number of rules. In the worst case, when we have isomorphic structures, the number of rules is exponential in the number of aligned nodes. Even though up to an exponential number of rules is produced by our definitions of consistency, we provide an algorithm for extracting all rules consistent with the node alignment that is $O(a^2 log\ a)$ in computational complexity, where a is the number of aligned nodes. We also provide a method of storing all consistent rules in a linear size data structure. In the following section we describe the method we use to efficiently extract and store large numbers of transfer rules.

3.4 Packed Transfer Rules

Deep syntactic structures can be stored as a set of constraints with each node of the structure labeled by an index number. Figures 3.10(a) and 3.10(b) show two example LFG f-structures for the sentence pair *"Sprachen spiegeln die Vielfalt der Europäischen Union wider"* and *"Languages reflect the diversity of the European Union"* displayed as AVMs and graph structures, respectively, and Figure 3.10(c) shows the pair encoded as two sets of constraints.[6]

Deep syntax transfer rules can be encoded in a similar way by encoding each side of the rule as a set of constraints, but instead of labeling the nodes with index numbers, each pair of aligned nodes is labeled with a variable, X_i. Figure 3.12 shows the transfer rule in Figure 3.11(f) encoded as two sets of constraints.

For each training pair of deep syntactic structures there can exist up to an exponential number of consistent transfer rules. Our method of packing transfer rules takes advantage of the fact that multiple rules extracted from the same deep syntactic structure pair will have constraints in common. For example, Figure 3.11(a)

[6] Atomic features are not shown.

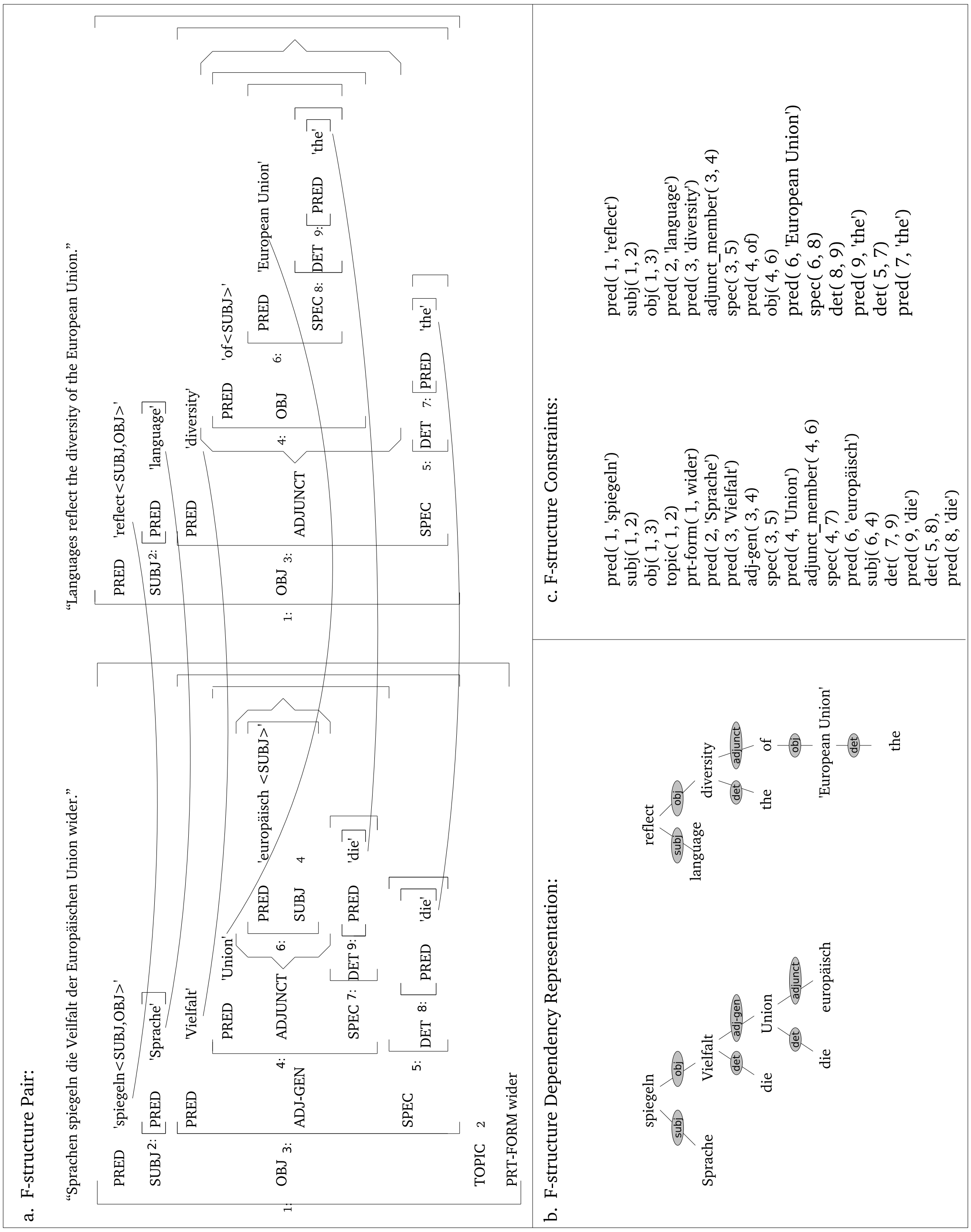


Figure 3.10: (a) LFG f-structures for the sentence pair *“Sprachen spiegeln die Vielfalt der Europäischen Union wider.”* and *“Languages reflect the diversity of the European Union”* as (a) AVMs, (b) graph structures and (c) constraints.

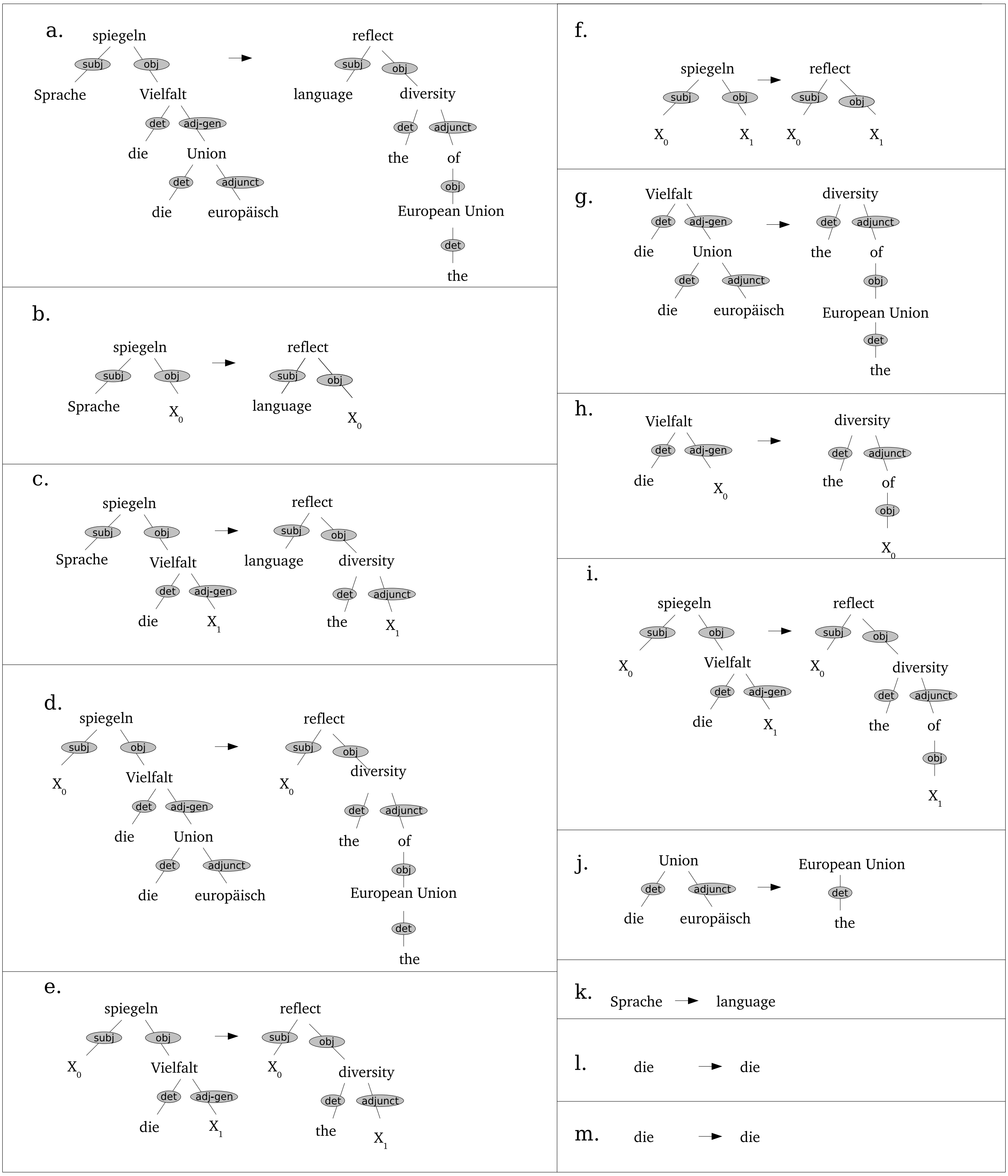


Figure 3.11: Example Transfer Rules: A subset of the transfer rules automatically extracted from training f-structure pair shown in Figure 3.10.

```
pred(X1,spiegeln)        pred(X1,reflect)
subj(X1,X2)          →   subj(X1,X2)
obj(X1,X3)               obj(X1,X3)
```

Figure 3.12: Example constraint-based encoding for transfer rule of Figure 3.11(f)

```
pred(X1,‘spiegeln’)
subj(X1,X2)                        pred(X1,‘reflect’)
obj(X1,X3)                         subj(X1,X2)
topic(X1,X2)                       obj(X1,X3)
prt-form(X1,wider)                 pred(X2,‘language’)
pred(X2,‘Sprache’)                 pred(X3,‘diversity’)
pred(X3,‘Vielfalt’)                det(X3,X4)
det(X3,X4)                   →     adjunct(X3,X8)
adj-gen(X3,X5)                     pred(X4,‘the’)
pred(X4,‘die’)                     pred(X8,‘of’)
pred(X5,‘Union’)                   obj(X8,X5)
det(X5,X6)                         pred(X5,‘European Union’)
adjunct(X5,X7)                     det(X5,X6)
pred(X7,‘europäisch’)              pred(X6,‘the’)
pred(X6,‘die’)
```

Figure 3.13: Constraints encoding transfer rule of Figure 3.11(a)

shows a transfer rule that maps the entire SL structure to the entire TL structure and Figure 3.13 shows the constraints that encode the rule.[7] Every other rule extracted from the same deep syntactic structure pair will consist of a subset of this set of constraints and storing each subsequent rule separately will therefore involve duplicating the constraints already recorded for this rule. Since the number of transfer rules that can be extracted from a given deep syntactic structure pair is exponential in the number of aligned nodes, storing transfer rules by enumerating each rule separately is inefficient. All consistent transfer rules can be packed into a single structure in which each constraint of the training pair of structures is only recorded once reducing the amount of space required from exponential to linear in the number of nodes.

[7] Atomic feature constraints are not shown.

3.4.1 Packed Transfer Rule Data Structure

The packed transfer rule data structure stores all rules consistent with the node alignment that are extracted from the same training deep syntactic structure pair in a single structure. We adopt a method of encoding, contextualized constraints, used in LFG parsing to improve the efficiency of processing disjunctive constraints of a grammar, that simplify the encoding of grammatical possibilities by allowing disjunctive statements as constraints (Maxwell and Kaplan, 1991). For example, the following disjunctive constraint, taken from Maxwell and Kaplan (1991), encodes the fact that the value of the atomic feature *case* for the German word *die* can be either nominative or accusative:

$$\text{case}(\ \textit{die}, \text{nom}) \vee \text{case}(\ \textit{die}, \text{acc})$$

When a sentence, to which this rule of the grammar applies, is parsed the packed f-structure representation will encode the two possibilities for the word *die* as contextualized constraints with an additional statement that signifies an exclusive disjunctive relation between the constraints labeled by the context variables. For example, Figure 3.14 shows the constraints for the German word *die* where the context variables, $A2$ and $A3$, label the two alternate constraints for *case* with the relation between the two constraints encoded by *choice([A2,A3],1)*. To extract a single f-structure from the packed representation a binary value is assigned to these context variables, where 1 signifies that the constraint labeled by the context variable *is included in* the extracted f-structure and 0 signifies the contrary, that the constraint *is not included in* the extracted f-structure, and the presence of `choice([A2,A3],1)` restricts the possible combinations of values to $\{A2 = 1, A3 = 0\}$ and $\{A2 = 0, A3 = 1\}$.

In a similar way, we use contextualized constraints to encode all transfer rules extracted from the same training structure pair by defining a context for a packed rule that determines under what circumstances each constraint is included or excluded from a given transfer rule. The entire set of SL constraints forms the LHS of

```
pred(A1,0,die)
case(A2,0,nom)
case(A3,0,acc)
choice([A2,A3], 1)
```

Figure 3.14: Contextualized constraints in f-structure encoding the possibility of *case* having value *nominative* or *accusative* for the German word *“die”*

```
pred( A0,X1,‘spiegeln’)
subj( A0,X1,X2)                    pred( A0,X1,‘reflect’)
obj( A0,X1,X3)                     subj( A0,X1,X2)
topic( A0,X1,X2)                   obj( A0,X1,X3)
pred( A1,X2,‘Sprache’)             pred( A1,X2,‘language’)
pred( A2,X3,‘Vielfalt’)            pred( A2,X3,‘diversity’)
det( A2,X3,X4)                     det( A2,X3,X4)
adj-gen(A2,X3,X5)            →     adj-mem( A2,X3,X8)
pred( A3,X4,‘die’)                 pred( A3,X4,‘the’)
pred( A4,X5,‘Union’)               pred( A2,X8,‘of’)
det( A4,X5,X6)                     obj( A2,X8,X5)
adj( A4,X5,X7)                     pred( A4,X5,‘European Union’)
pred( A5,X6,‘die’)                 det( A4,X5,X6)
pred( A4,X7,‘europäisch’)          pred( A5,X6,‘the’)
```

Figure 3.15: Contextualized constraints encoding all consistent transfer rules for deep syntactic structure of Figure 3.16

the packed rule, and the entire set of constraints of the TL structure forms the RHS. For example, Figure 3.15 shows the constraints that encode the packed transfer rule of the deep syntactic structure pair of Figure 3.10. Each pair of nodes that forms the root of a consistent transfer rule is assigned a single unique context variable, A_i, which is used to label all the constraints belonging to that node. For example, the nodes in bold typeface in Figure 3.16 are root nodes of consistent transfer rules and are assigned the context variables $A0$-$A5$. For a node that is not a rule root, its constraints are assigned the context variable of the closest rule root above it in the structure. For example, the node *europäisch* in Figure 3.16 is assigned the context variable $A4$, the context variable of its head, since it is not a rule root itself.

Extracting a particular transfer rule from the packed structure now simply involves assigning the value *1* to the constraints included in the rule and *0* to the constraints

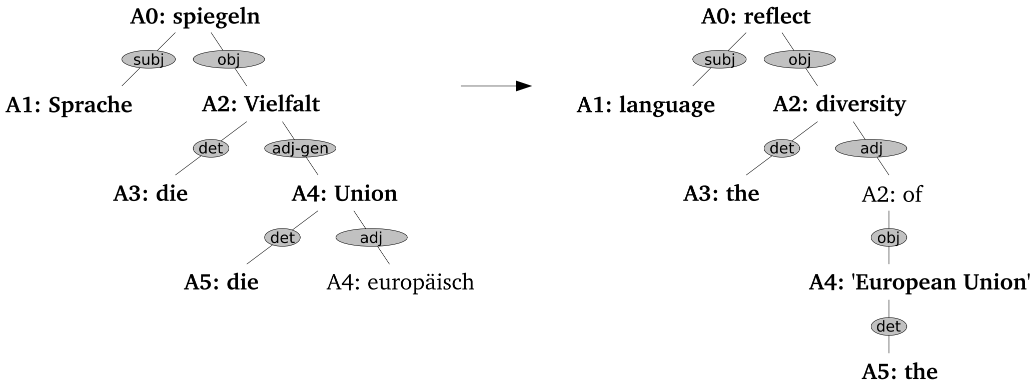


Figure 3.16: Packed Transfer Rule with Context Variables

that are excluded from the rule. Figure 3.17 shows one of the possible combinations of values for the set of context variables given to the constraints of the packed rule shown in Figure 3.16 and the rule that results by taking the constraints labeled *1* for this particular combination of boolean values.

As described so far, encoded in the packed structure is a superset of the set of consistent transfer rules, as the packed structure also contains discontiguous rules disallowed by Definitions 1 and 2. We eliminate discontiguous rules by only allowing context variables of contiguous parts of the structure to be assigned the value 1.

3.4.2 Transfer Rule Extraction Algorithm

Transfer rule extraction works by encoding all consistent rules extracted from a pair of deep syntactic structures in the packed representation described in Section 3.4.1. The main part of the algorithm decides which pairs of nodes within the deep syntactic structure pair form *rule roots*. Once the *rule roots* have been determined, the entire set of SL and TL constraints are then simply recorded with each constraint labeled with a *context variable*, A_i, in addition to replacing the original structure node labels with variables (X_i). The algorithm for choosing the *rule root* nodes of the deep syntactic structure pair is given in Figure 3.18. In our implementation, we use some of Prolog's built-in features that are not available in other programming languages. Therefore, in order to keep the pseudo code in Figure 3.18 as implemen-

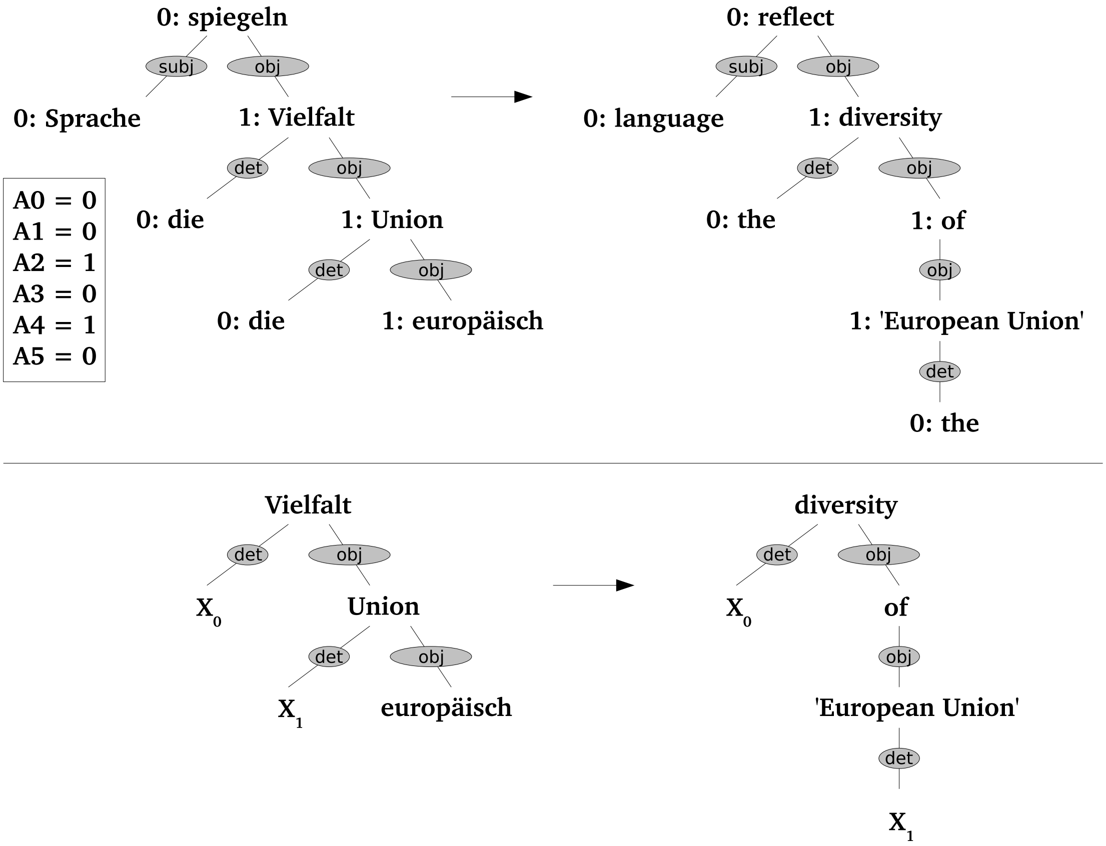


Figure 3.17: Packed transfer rule assignment of values to context variables and resulting transfer rule

tation independent as possible, when we use a Prolog specific function or control structure we describe it in pseudo code as an equivalent function or control structure available in most programming languages. For example, Prolog has a built in indexing of terms, that uses the first argument of the term as a key to achieve an O(log n) return time when searching for that term in memory. We use this in our Prolog implementation but where we do so we describe it in pseudo code as a hash table. The complexity of the algorithm is $O(a^2 log\ a)$ in the worst case, where a is the number of aligned node pairs.

3.5 Source Language Back-off Transfer Rules

When translating unseen data, words that are out of coverage of the transfer rules extracted from the bilingual training data are likely to be encountered. To handle out of coverage words we use *source language back-off transfer rules.* Figure 3.19 shows an example German structure and Figure 3.20 shows an example rule for translating the (out of coverage) German word *signalisieren.* A source language back-off transfer rule translates a single SL node to the target language as itself, so that during decoding full transfer rule coverage of the SL dependency graph can be assumed. The arguments of the SL node are transferred to one of two possible places in the TL structure. If the SL dependency relation exists in at least one parse of the TL side of the bitext corpus, it is assumed to be a valid dependency relation for the TL grammar and the dependency relation is transferred as-is to the TL structure. If, on the other hand, the dependency relation has never been seen in the TL corpus, we preserve the relation between the two nodes but label it with the most frequent dependency relation in the TL side of the corpus.

```
Algorithm RuleRoots( List sl_nodes, List tl_nodes,
                     HashTable <sl_node_id,alignment_id> sl_alignments,
                     HashTable <tl_node_id,alignment_id> tl_alignments):

  # For each aligned SL node create a list
  # containing the alignment ids of its aligned descendents
  # Put lists in a Hash Table
  sl_aligned_descs = new HashTable<list_of_aligned_descs,sl_node_id>
  foreach s ∈ S
    if exists sl_alignments.get(s.node_id) then
      list = new empty list
      foreach d ∈ descendents(s)
        if exists sl_alignments.get( d.node_id) then
          a_id = sl_alignment.get( d.node_id)
          list.add( a_id)
      sl_aligned_descs.put( list, s)

  # Likewise for TL nodes
  tl_aligned_descs = new HashTable<list_of_aligned_descs,tl_node_id>
  foreach t ∈ T
    if exists tl_alignments.get(t.node_id) then
      list = new empty list
      foreach d ∈ descendents(t)
        if exists tl_alignments.get( d.node_id) then
          a_id = tl_alignment.get( d.node_id)
          list.add( a_id)
      tl_aligned_descs.put( list, t)

  # Find node pairs with matching sets of aligned descendents
  roots = new empty List
  foreach key in keys( sl_aligned_desc)
    if exists tl_aligned_descs.get( key)
      # A pair has been found
      i = sl_aligned_descs.get( key)
      j = tl_aligned_descs.get( key)
      roots.add( i, j)
  return roots
```

Figure 3.18: Algorithm to choose the *rule roots* in the SL and TL deep syntactic structures

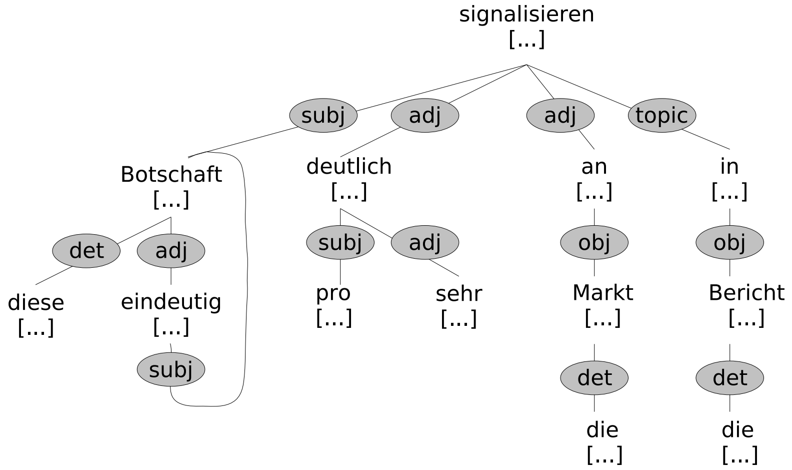


Figure 3.19: Example German deep syntactic structure for sentence *"Im Bericht wird diese eindeutige Botschaft an den Markt sehr deutlich signalisiert."*

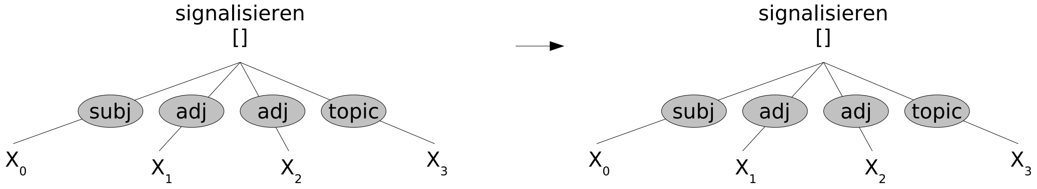


Figure 3.20: Example source language back-off transfer rule for German word *signalisieren*

3.6 Conclusion

In this chapter, we described a method of automatically extracting transfer rules from pairs of deep syntactic structures, by automatically aligning the structures at node level before extracting all rules consistent with the node alignment. We provided a definition of consistency that avoids erroneous transfer rules allowed in previous approaches and that do not rely on isomorphism between training pairs of structures. In addition, we described efficient methods of extracting and storing large numbers of rules. The transfer rule extraction tool and source code were made available in order to aid future research (Graham and van Genabith, 2009). In the chapters that follow we describe how the rules described in Sections 3.2, 3.3 and 3.4 that are automatically extracted from the training corpus are used to compute the translation model (Chapter 4) and how they are used to translate SL deep syntactic structures into the TL during transfer decoding (Chapter 5).

Chapter 4

Deep Syntax Translation Model

4.1 Introduction

In this chapter, we describe in detail our deep syntax translation model. In our model, we use both *decoding features*, applied to TL hypothesis structures as they are constructed during decoding, and *post-generation features* applied after generation to the TL surface form sentences output by the generator. Decoding features include a translation model computed from relative frequencies of transfer rules automatically extracted from the training corpus, as well as a deep syntax language model trained on monolingual TL data, among others. The post-generation features include a surface form/string-based language model and a grammaticality feature that uses information output by the generator about the grammaticality of generated translations.

4.2 Translation Model

As in PB-SMT, a deep syntax translation model can be defined as a log-linear combination of several feature functions:

$$p(e|f) = exp\sum_{i=1}^{n}\lambda_i h_i(e,f)$$

All but two of the feature functions we use are applied to the deep syntactic structure during decoding with two final feature functions, the surface form language model and a grammaticality feature, both applied after decoding to generated strings. Weights are optimized on a development set using Minimum Error Rate Training (Och, 2003).

4.3 Deep Syntax Translation Model

In PB-SMT the translation of an input sentence into an output sentence is modeled by breaking down the translation of the sentence into the translation of a set of phrases (Koehn et al., 2003). Similarly, for deep syntactic transfer-based SMT, the transfer of the SL structure **f** into a TL structure **e** can be broken down into the transfer of a set of rules $\{\bar{f}, \bar{e}\}$:

$$p(\bar{f}_1^I | \bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i | \bar{e}_i)$$

We compute all rules from the training corpus and estimate the translation probability distribution by relative frequency of extracted transfer rules:

$$\phi(\bar{f} | \bar{e}) = \frac{count(\bar{e}, \bar{f})}{\sum_{\bar{f}_i} count(\bar{e}, \bar{f}_i)}$$

This is carried out in both the source-to-target and target-to-source language directions.

4.3.1 Counting Rules

All transfer rules consistent with the word alignment are extracted yielding large numbers of rules stored efficiently in a packed representation (Graham and van Genabith, 2008). When counting transfer rules to compute the translation model, the question arises of how we should deal with atomic features and values within transfer rules, i.e. should two transfer rules that contain identical lexical items and dependency relations but different atomic feature values be treated as two instances of the same rule or as two distinct rules? When we compute the translation model we ignore atomic features and their values, so two such rules are treated as instances of the same rule. Similar to Factored Models (Koehn and Hoang, 2007), excluding atomic features/morphological factors when computing the translation model for lemmas results in a statistically richer model. Figure 4.1 shows four example transfer rule tokens for translating the German word *Katze* to *cat* that are all considered to be an instance of the same rule type. This decision is motivated by the large number of possible values of atomic features in transfer rules, $O(v^f)$, where f is the number of atomic features and v is the number of possible values for a feature. A verb in a LFG f-structure, for example, usually has the atomic features and possible values shown in Figure 4.2. The number of possible combinations of atomic feature

$$\begin{array}{ccc} \text{Katze} & \rightarrow & \text{cat} \\ \begin{bmatrix} \text{common} & \text{mass} \\ \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{sg} \\ \text{gend} & \text{fem} \\ \text{case} & \text{nom} \end{bmatrix} & & \begin{bmatrix} \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{sg} \\ \text{case} & \text{nom} \end{bmatrix} \end{array}$$

$$\begin{array}{ccc} \text{Katze} & \rightarrow & \text{cat} \\ \begin{bmatrix} \text{common} & \text{mass} \\ \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{sg} \\ \text{gend} & \text{fem} \\ \text{case} & \text{nom} \end{bmatrix} & & \begin{bmatrix} \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{sg} \\ \text{case} & \text{nom} \end{bmatrix} \end{array}$$

$$\begin{array}{ccc} \text{Katze} & \rightarrow & \text{cat} \\ \begin{bmatrix} \text{common} & \text{mass} \\ \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{pl} \\ \text{gend} & \text{fem} \\ \text{case} & \text{nom} \end{bmatrix} & & \begin{bmatrix} \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{pl} \\ \text{case} & \text{nom} \end{bmatrix} \end{array}$$

$$\begin{array}{ccc} \text{Katze} & \rightarrow & \text{cat} \\ \begin{bmatrix} \text{common} & \text{mass} \\ \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{sg} \\ \text{gend} & \text{fem} \\ \text{case} & \text{acc} \end{bmatrix} & & \begin{bmatrix} \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{sg} \\ \text{case} & \text{obl} \end{bmatrix} \end{array}$$

Figure 4.1: Example transfer rule token instances of a single rule type

Atomic Feature	Values
verb-type	aux, copular, main, modular, noncopular, predicative, raising
passive	+,-
clause-type	adv, cond, decl, imp, in, nom, pol-int, rel, wh-int
tense	past, pres, fut
progressive	+, -
perfect	+, -
mood	imperative, indicative, subjunctive, successive

Figure 4.2: Atomic features and values for verbs in LFG

values for a verb in an LFG f-structure is therefore 6048 and a transfer rule that translates a German verb into an English verb has 6048^2 possible combinations of atomic feature values.

It's worth mentioning how this relates to other SMT approaches. For argument's sake, consider the case, where we (somehow) already know the lemmas, dependency relations and atomic features (but not their values) of the deep syntactic structure of a fluent and adequate translation and only need to assign the correct values to the atomic features. The equivalent case for Phrase-Based Factored Models is when it (somehow) already knows the correct TL lemmas and only needs to generate the correct surface form inflection for each lemma. The task facing the deep syntax approach is much more difficult, due to the far greater number of possible sets of atomic values compared to the number of possible surface form inflections. For example, the English verb *overlook* has four possible surface form inflections (*overlook*, *overlooks*, *overlooking* and *overlooked*), and 6048 possible sets of atomic feature values, a selection of which are shown in Figure 4.3. In addition, the problem of guessing the correct combination of TL atomic features is exacerbated by the fact that in SMT via deep syntactic transfer, we are restricted to sentence-level generation, as opposed to the word-level generation of Phrase-Based Factored Models, and this imposes harsh limits on the number of translation options that can be generated greatly increasing the possibility of early pruning of good translations.

Example	Tense	Prog	Perf	Mood	Pass	Clause-type
John **overlooks** Mary.	pres	-	-	ind	-	decl
John **overlooked** Mary.	past	-	-	ind	-	decl
John will **overlook** Mary.	fut	-	-	ind	-	decl
John has **overlooked** Mary.	pres	-	+	ind	-	decl
John will have **overlooked** Mary.	fut	-	+	ind	-	decl
Has John **overlooked** Mary?	pres	-	+	ind	-	int
Will John have **overlooked** Mary?	fut	-	+	ind	-	int
John is **overlooked**.	pres	-	-	ind	+	decl
Is John **overlooked**?	pres	-	-	ind	+	int
John will be **overlooked**.	fut	-	-	ind	+	decl
Will John be **overlooked**?	fut	-	-	ind	+	int
John was **overlooked**.	past	-	-	ind	+	decl
Was John **overlooked**?	past	-	-	ind	+	int
John has been **overlooked**.	pres	-	+	ind	+	decl
Has John been **overlooked**?	pres	-	+	ind	+	int
John will have been **overlooked**.	fut	-	+	ind	+	decl
Will John have been **overlooked**?	fut	-	+	ind	+	int
John was being **overlooked**.	past	+	-	ind	+	decl
Was John being **overlooked**?	past	+	-	ind	+	int
...						

Figure 4.3: A selection of atomic feature values and surface form morphological inflections for *overlook*

4.4 Lexical Weighting

In PB-SMT, lexical weighting is used as a back-off to the translation model since it provides richer statistics and therefore more reliable probability estimates. Adapting this feature to deep syntax is straightforward. In PB-SMT the lexical translation probability of a phrase pair is computed based on the alignment between the words in the phrase pair. For deep syntactic transfer, we compute the lexical translation probability instead using the alignment of lexical items in the LHS and RHS of a transfer rule. The lexical translation probability of a RHS, $\bar{e}$, given the LHS, $\bar{f}$ and alignment a, is estimated as follows:

$$lex(\bar{e}|\bar{f},a) = \prod_{i=1}^{length(\bar{e})} \frac{1}{|\{j|(i,j) \in a\}|} \sum_{\forall (i,j) \in a} w(e_i|f_j)$$

We use lexical weighting in both language directions.

4.5 Language Model

The overall system employs a language model at two different stages; a trigram dependency-based language model is used during transfer decoding to model fluency of combinations of words in TL deep syntactic structures as they are constructed and a string-based trigram language model is used after generation to model fluency for generated translations.

4.5.1 Deep Syntax Language Model

In PB-SMT, ngram language models are used to produce fluent translations, where the decoder output is a sequence of surface form words. In deep syntactic transfer, the output of the decoder is a deep syntactic structure with words organized in the form of a graph instead of a linear sequence. A string-based language model cannot be used during transfer decoding because no surface form representation of the TL deep syntactic structure is available.

It is still important, however, for the model to take TL fluency into account so that the structures it outputs contain fluent combinations of words. Figure 4.4 shows part of a deep syntactic structure for a German sentence that contains the phrase *Bewusstsein für Dringlichkeit*. When translating into English, if, for example, three different transfer rules were used to translate each of the German words and TL fluency was not included in the model, the system would rank English Structure 1 in Figure 4.4, *sense for urgency*, higher than the more fluent English Structure 2, *sense of urgency* since the transfer rule *für* → *for* has a higher probability than *für* → *of*. If a deep syntax language model is used, however, the more fluent combination of words, *sense of urgency*, should have a higher language model probability than its less fluent counterpart.

In Section 2.3.3 we described how a standard language model computes the probability of a sequence of English words by combining the probability of each

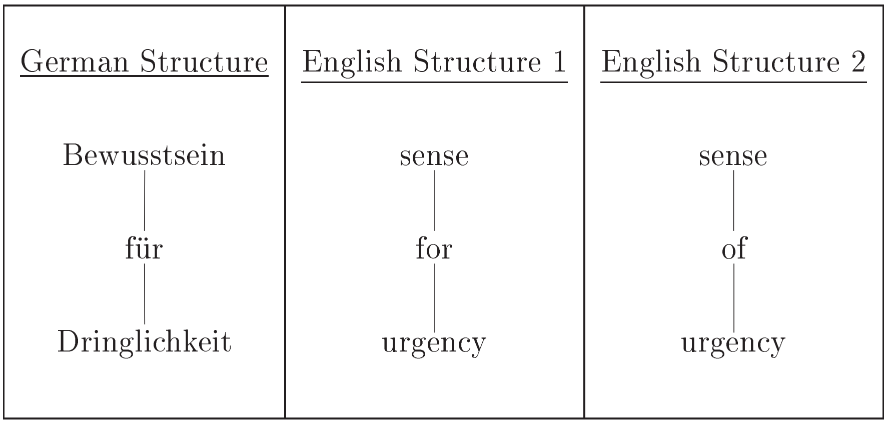


Figure 4.4: German Deep Syntactic Structure for *Bewusstsein für Dringlichkeit* and two possible translations into English *sense for urgency* and *sense of urgency*

word, w_i, in the sequence given the preceding sequence of $i-1$ words.

$$p(w_1, ...w_l) = \prod_{i=1}^{l} p(w_i|w_1, ..., w_{i-1}) \tag{4.1}$$

In a similar way, we can compute the probability of a deep syntactic structure, d, with root node, w_r, consisting of l words, by combining the probability of each word, w_i, in the structure given the sequence of words linked to it via dependency relations from root word, w_r, to word $w_{m(i)}$, where function m, maps the index of a non-root word to the index of its head node within the structure:

$$p(d) = \prod_{i=1}^{l} p(w_i|w_r, ..., w_{m(m(i))} w_{m(i)}) \tag{4.2}$$

Figure 4.5(a) shows an example of how Equation 4.1 can be used to compute the probability of the English sentence *"I saw the red house"* with a standard language model and Figure 4.6(d) shows an example of how Equation 4.2 can be used to compute the probability of the deep syntactic structure, shown in Figure 4.6(c), for the same sentence.

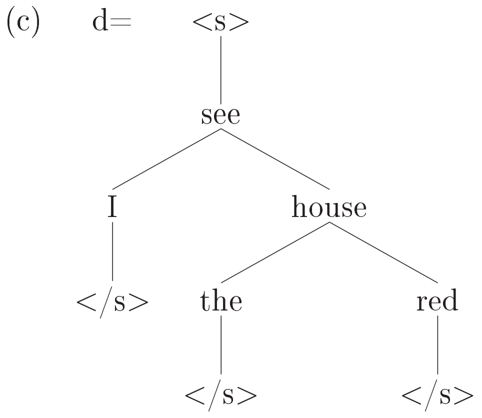
In order to combat data sparseness, we apply the Markov assumption, as is done in standard language modeling, and simplify the probability of a deep syntactic structure by only including a limited length of history when computing the proba-

String-based Language Model

$$
\text{(a)}\quad p(\text{I saw the red house}) = \begin{array}{l} p(\text{I}|\,\text{<s>}) \\ p(\text{saw}|\,\text{<s> I}) \\ p(\text{the}|\,\text{<s> I saw}) \\ p(\text{red}|\,\text{<s> I saw the}) \\ p(\text{house}|\,\text{<s> I saw the red}) \\ p(\text{</s>}\,|\,\text{<s> I saw the red house}) \end{array}
$$

$$
\text{(b)}\quad p(\text{I saw the red house}) = \begin{array}{l} p(\text{I}|\,\text{<s>}) \\ p(\text{saw}|\,\text{<s> I}) \\ p(\text{the}|\text{I saw}) \\ p(\text{red}|\text{saw the}) \\ p(\text{house}|\text{the red}) \\ p(\text{</s>}\,|\text{red house}) \end{array}
$$

Deep Syntax Language Model



$$
\text{(d)}\quad p(\text{d}) = \begin{array}{l} p(\text{see}|\,\text{<s>}) \\ p(\text{I}|\,\text{<s> see}) \\ p(\text{</s>}\,|\,\text{<s> see I}) \\ p(\text{house}|\,\text{<s> see}) \\ p(\text{the}|\,\text{<s> see house}) \\ p(\text{</s>}\,|\,\text{<s> see house the}) \\ p(\text{red}|\,\text{<s> see house}) \\ p(\text{</s>}\,|\,\text{<s> see house red}) \end{array}
$$

$$
\text{(e)}\quad p(\text{d}) = \begin{array}{l} p(\text{see}|\,\text{<s>}) \\ p(\text{I}|\,\text{<s> see}) \\ p(\text{</s>}\,|\,\text{see I}) \\ p(\text{house}|\,\text{<s> see}) \\ p(\text{the}|\,\text{see house}) \\ p(\text{</s>}\,|\,\text{house the}) \\ p(\text{red}|\,\text{see house}) \\ p(\text{</s>}\,|\,\text{house red}) \end{array}
$$

Figure 4.5: Standard Language Model and Deep Syntax Language Model Example for *"I saw the red house"*

bility of each word in the structure. A trigram deep syntax language model combines the probability of each word in the structure given *the head of the head of the word* and *the head of the word* in the structure:

$$p(d) = \prod_{i=1}^{l} p(w_i | w_{m(m(i))}, w_{m(i)}) \tag{4.3}$$

Figure 4.5(b) and 4.5(e) show examples of how the Markov assumption is applied in a standard trigram language model and a deep syntax trigram language model, respectively.

Additional Simplification for Argument Sharing

Argument sharing can occur within deep syntactic structures and in such cases we use a simplification of the actual deep syntax graph structure by introducing the restriction that each word in the structure may have a single head word (with the exception of the root word which has no head word), as this is required for the m function, that maps the index of each word to that of its head word. Figure 4.6 shows an example of the underlying graph structure for the LFG f-structure for the English sentence *"The cat likes to sleep"* in which the subject of both *like* and *sleep* is *cat*. When computing the probability estimate of the structure we ignore the fact that *cat* is an argument of *sleep*.

Dependency Relations

In our approach, we chose to omit dependency relations from our language model. The motivation behind this was mainly to keep the approach as similar to standard PB-SMT as possible. However, since a deep syntax architecture provides information about dependency relations between TL words, it could well prove advantageous to include these relations in a language model, so that more fluent combinations of lexical items and dependency relations can be ranked higher by the model than less fluent ones. Due to time constraints, however, we leave further investigation into

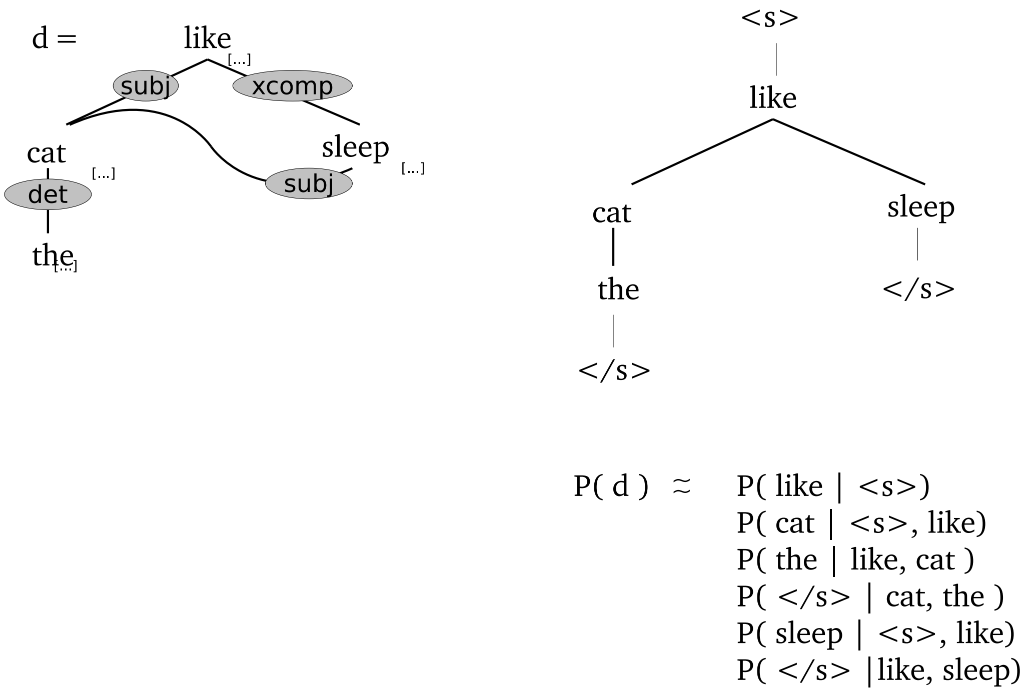


Figure 4.6: Example of structure simplification for deep syntax language modeling when argument sharing occurs

this topic as future work.

4.6 Penalty Features

We use a *word penalty* feature that adds a factor, w, for each word in the target language structure in order to counteract the bias of the deep syntax language model toward smaller output structures.

For similar reasons, we use a *rule penalty feature*. The translation model will, in general, prefer smaller rules, since they occur more frequently in the training data, but since larger rules are more likely to produce fluent combinations of words we use a feature that allows the system to take the number of rules used to construct a target language structure into account, so all other things being equal it can prefer a hypothesis structure that was constructed using fewer rules.

In addition to this, the number of *fragments* in the TL structure are taken into account. The decoder can produce a fragmented target structure if it uses a transfer rule that was extracted from a training example pair in which the TL training structure was itself fragmented. We use a *fragment* penalty that adds a factor for each fragment in the target structure so that translations generated from structures with more fragments can be dispreffered, as they are likely to lead to ungrammatical translations.

In addition, a penalty feature is used to allow the system to disprefer translations produced using source language back-off transfer rules. As described in Section 3.5, these back-off rules are used to translate out of coverage words. We include a *source-language back-off rule penalty feature* that introduces a factor for every rule of this type used to produce a translation.

4.7 Atomic Feature Match

For high coverage of transfer rules on unseen data, the atomic features and values of a source language structure are not required to match those of a transfer rule in order for the rule to be used for translation. Figure 4.7 shows an example transfer rule and source language structure in which there is a mismatch between the value of the atomic feature *case*. We allow the rule to be used to translate the structure, but the fact that an atomic feature did not match is taken into account so that (all else being equal) translations produced by a rule with more atomic feature values matching those of the SL structure can be preferred. There are two reasons we do this.

Firstly, when an atomic feature value of a rule does not match the source structure, the TL value of that feature is translated separately from the lemma (discussed in detail in Section 5.5). For example, in Figure 4.7 since the value of *case* does not match that of the transfer rule, the value of *case* in the target structure is translated separately from *Handel*. Translating feature values separately from lemmas increases the likelihood of the target language output no longer being fluent, and since the particular combination of lemmas and atomic feature values may not have

Transfer Rule

$$\begin{array}{ccc} \text{Handel} & \rightarrow & \text{trade} \\ \begin{bmatrix} \text{common} & \text{mass} \\ \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{sg} \\ \text{gend} & \text{masc} \\ \text{case} & \textbf{acc} \end{bmatrix} & & \begin{bmatrix} \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{sg} \\ \text{case} & \text{obl} \end{bmatrix} \end{array}$$

Source Language Structure

$$\begin{array}{c} \text{Handel} \\ \begin{bmatrix} \text{common} & \text{mass} \\ \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \text{sg} \\ \text{gend} & \text{masc} \\ \text{case} & \textbf{nom} \end{bmatrix} \end{array}$$

Figure 4.7: Atomic feature value mismatch example when translating *"Handel"* into English

been observed in the TL training data.

Secondly, atomic features and their values can sometimes be useful for guiding translation. All else being equal, the more atomic feature values of a rule that match the SL structure the more likely it is to produce an adequate translation. For example, Figure 4.8 shows two transfer rules that translate the German word *Haar* firstly as *hair* and secondly as *strand*. When *Haar* appears in the singular in German, the English word *strand* is a possible adequate translation, but when *Haar* occurs in the plural in German, *strand* is no longer an adequate translation for *Haar*, but rather *hair* is.[1] By including a feature that takes the number of matching atomic feature and values into account, the system can prefer Transfer Rule 2 over

[1] *Das ist ein langes Haar* = *It's a long hair* = *It's a long strand*; *Er schneidet die Haare* = *He cuts hair* ≠ *He cuts strands*, as *strands* is not used in this context in English.

Transfer Rule 1 (all else being equal) when translating the source language structure in Figure 4.8.

We define the *source atomic value match feature* function, s, where S is the set of atomic feature-value pairs in the source language structure and R is the set of atomic feature-value pairs of the set of transfer rules as:

$$s(S, R) = \frac{|S \cap R|}{|S|} \tag{4.4}$$

When matching transfer rules to source language structure we also need to take into account the fact that there may be a different number of atomic features in S as opposed to R. For example, the LFG grammars we currently use for parsing produce different sets of atomic features for the same lemma in different situations. For example, a verb may or may not have the atomic feature *perfect* depending on its value for the atomic feature *tense*. Since we know in many cases $|S|$ is not equal to $|R|$, we include an additional feature, that takes into account the total number of atomic features in the transfer rules. The *rule atomic match* feature function, r, is as follows:

$$r(S, R) = \frac{|S \cap R|}{|R|} \tag{4.5}$$

4.8 Post-generation Features

4.8.1 Surface Form/String-based Language Model

After generation we apply a standard language model trained on the surface form TL data to estimate the probability for each generated string. The surface form language model is used as a feature in the overall model so that all else being equal a translation with a higher surface form language model probability can be preferred.

Transfer Rule 1

$$\text{Haar} \quad \rightarrow \quad \text{hair}$$

$$\begin{bmatrix} \text{common} & \text{mass} \\ \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \textbf{pl} \\ \text{gend} & \text{neut} \\ \text{case} & \text{nom} \end{bmatrix} \qquad \begin{bmatrix} \ldots \end{bmatrix}$$

Transfer Rule 2

$$\text{Haar} \quad \rightarrow \quad \text{strand}$$

$$\begin{bmatrix} \text{common} & \text{mass} \\ \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \textbf{sg} \\ \text{gend} & \text{neut} \\ \text{case} & \text{nom} \end{bmatrix} \qquad \begin{bmatrix} \ldots \end{bmatrix}$$

Source Language Structure

Translations

$$\text{Haar} \quad \rightarrow \quad \underset{\text{(adequate)}}{\text{hair}} \quad \text{or} \quad \underset{\text{(inadequate)}}{\text{strand}}$$

$$\begin{bmatrix} \text{common} & \text{mass} \\ \text{nsyn} & \text{common} \\ \text{pers} & 3 \\ \text{num} & \textbf{pl} \\ \text{gend} & \text{neut} \\ \text{case} & \text{nom} \end{bmatrix} \qquad \begin{bmatrix} \ldots \end{bmatrix} \qquad \begin{bmatrix} \ldots \end{bmatrix}$$

Figure 4.8: Example of how atomic feature values can be used to guide translation

4.8.2 Grammaticality Feature

XLE uses a precision grammar, a set of rules for parsing/generating grammatical sentences and a back-off set of *fragmented* rules for robustness that allow ungrammatical sentences (or sentences outside the coverage of the precision grammar) to be parsed/generated. The generator firstly attempts to generate output for an input f-structure using the precision grammar rules. If it fails, due to a clash of constraints, for example if the *case* of a noun is incorrect or an argument of a verb is missing, it reverts to the fragment grammar and generates the output marked as *ungrammatical*. We use this marker as a binary feature in our model.

4.9 Conclusion

In this chapter, we presented the model used by the system to rank hypothesis translations. We described how we use a log-linear combination of several feature functions, most of which are applied to the deep syntactic structure during transfer decoding, such as deep syntax translation model and deep syntax language model, and two feature functions (the string-based language model and grammaticality feature) that are applied after generation to the surface form output string. In the following chapter we discuss how transfer decoding is carried out using the decoding features we just described.

Chapter 5

Transfer Decoding

5.1 Introduction

SMT via deep syntactic transfer is composed of three parts, (i) parsing to deep syntactic structure, (ii) transfer from SL structure to TL structure and (iii) generation of TL sentence. This chapter is mainly concerned with step (ii): a search for the n-best TL deep syntactic structures by means of transfer decoding. Transfer decoding carries out the task of constructing TL structures by applying transfer rules to the SL structure. The transfer decoder takes a single SL deep syntactic structure as input and applies the LHS of transfer rules to snippets of the SL structure. The RHS of transfer rules are combined to produce TL translation hypothesis structures. In this chapter, we firstly describe the top-down application of transfer rules, and our criteria for admissibility of transfer rules to decoding, followed by the heuristic search algorithm used to manage the large search space of TL structures. Next, we give details of how we integrate a deep syntax trigram language model into decoding, before finally describing how we translate atomic features using an adaptation of Factored Models. The source code of the decoder developed as part of this thesis was made available to assist the progress of future research (Graham, 2010).

5.2 Transfer Rule Application

Decoding takes a single SL structure as input and involves a search for the n-best TL structures. The decoding algorithm works by creating TL solutions via a top-down application of transfer rules to the SL structure beginning at the root. When the LHS of a rule unifies with the SL structure, the RHS produces a portion of TL structure. Figure 5.1 shows an example application of three rules to the deep syntactic structure for the German sentence *Die Katze schläft gern - The cat likes to sleep* shown in Figure 5.1(a). Figure 5.1(b) shows the first transfer rule applied to the root node of the SL structure producing the TL structure portion shown in Figure 5.1(c). Transfer rule variables map arguments in the SL structure to the desired position when creating a TL solution. For example, variable X_0 in

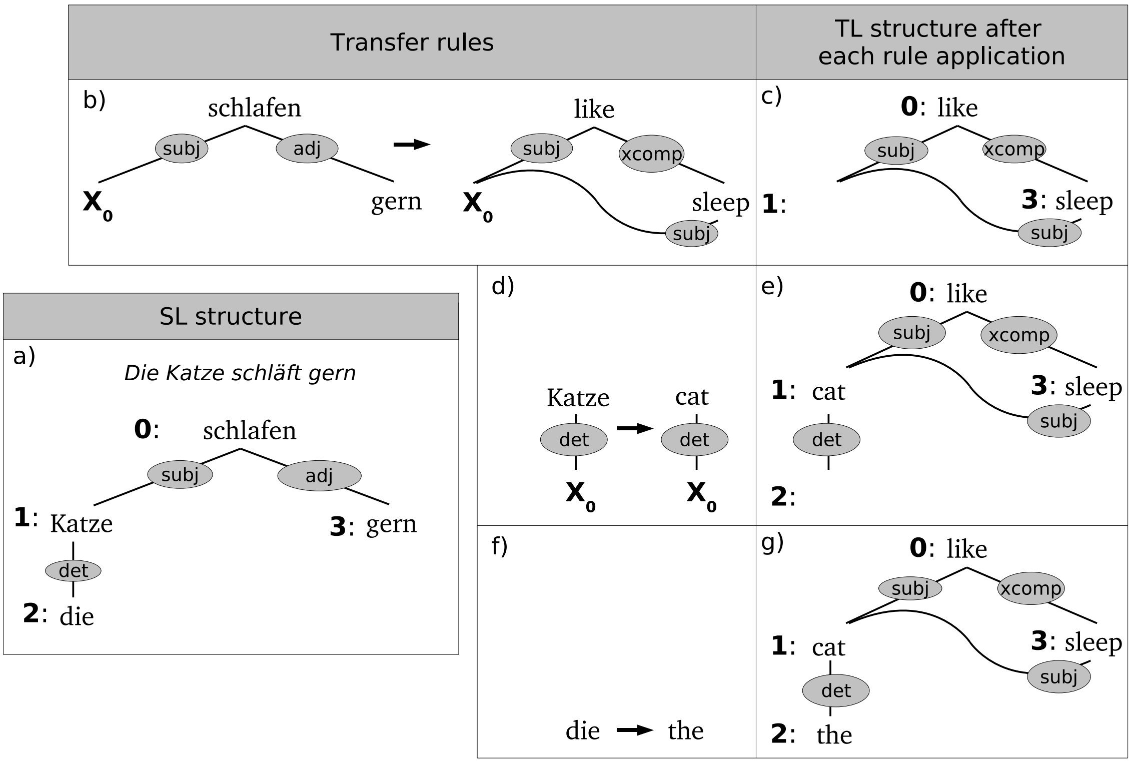


Figure 5.1: Example top-down application of transfer rules

Figure 5.1(b) maps the *subject* of *schlafen* to the *subject* of *like* in the TL structure labeled with id number *1* shown in Figure 5.1(c). Next, *Katze* in the SL structure is translated (Figures 5.1(d) and 5.1(e)), before finally, *die* is translated (Figures 5.1(f) and 5.1(g)).

5.2.1 Transfer Rule Decoder Admissibility Criteria

The LHS of transfer rules admitted to decoding are required to match a contiguous snippet of the SL *preds-only* structure, as this greatly reduces the likelihood of TL structures being fragmented. For example, when translating an intransitive instance of a verb that can be both intransitive and transitive, such as *lesen* (to read) in German for example, only transfer rules that have a *subject* argument are admitted to decoding, and similarly when translating a transitive instance, only rules containing *lesen* with both a *subject* and *object* are admitted. None of the atomic features or their values of a transfer rule are required to match that of the

SL structure in order for a transfer rule to be used for decoding, but, as described in Sections 4.3.1 and 4.7 we include a feature based on the number of matching atomic features in our model to allow the system to prefer solutions in which more atomic features of the LHS of transfer rules match those of the SL structure.

Recall, from Section 4.3.1, mainly for data sparseness reasons, due to the large number of possible combinations of atomic feature values in transfer rules, that when we identify rule types for computing the translation model, we do not use *full* transfer rule types, i.e. including atomic features, but rather *preds-only* transfer rule types, i.e. ignoring differences in any atomic features and their values. In keeping with this, for decoding, we only apply a single preds-only transfer rule type to the SL structure. This is not only motivated by the fact that this was the assumption when computing the translation model, but also because it greatly reduces the decoder search space, and allows it to focus on important differences in lexical choice rather than minor differences in values of atomic features. For example, Figure 5.2 shows three *full* transfer rule types all with the same preds-only structure. Only a single *preds-only* transfer rule type from these three rules will be admitted to decoding for the SL structure shown in Figure 5.1(a).[1] Even for this simple example, admitting only one of the three transfer rules to decoding reduces the number of TL hypothesis structures by a factor of three, and since the number of possible combinations of atomic features for a single preds-only rule type is very large (discussed in Section 4.3.1), the overall reduction in search space is considerable, allowing the search to focus on important differences, such as lexical choice and dependency relations, as opposed to differences in atomic feature values. Note that since we allow a fuzzy match of the kind described above between atomic features of transfer rules and SL structures, using the preds-only transfer rule type does not reduce coverage of unseen SL structures.

Of course, allowing a fuzzy match between atomic feature values of transfer rules and SL structures, will sometimes result in inadequate translations. We describe our

[1] The particular instance of the rule that is selected is an arbitrary choice.

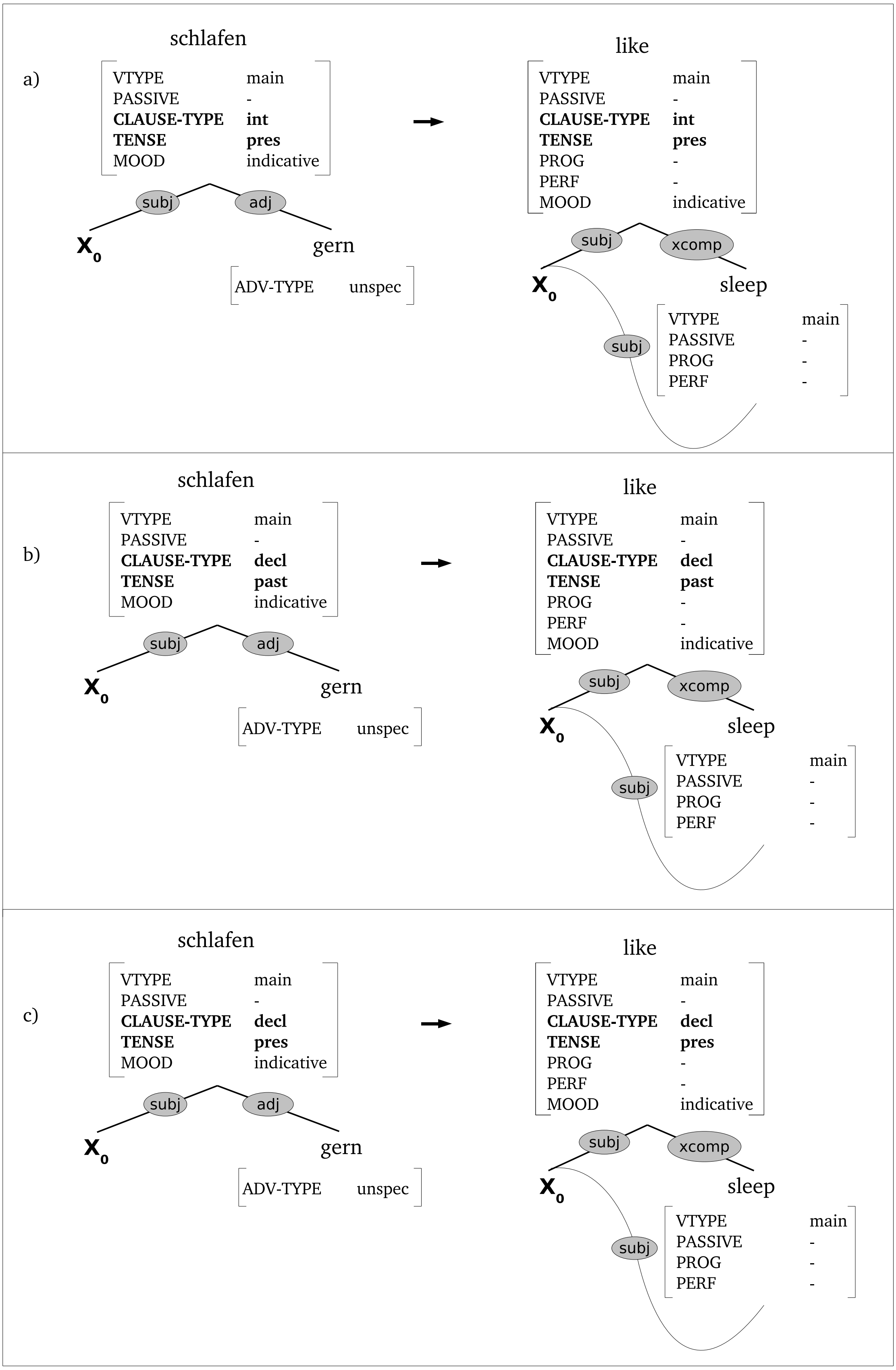


Figure 5.2: Full transfer rule types (a), (b) and (c) with different values for atomic features CLAUSE-TYPE and TENSE.

solution to this in Section 5.5.

5.3 Beam Search

Partial translations (or translation hypotheses) are constructed by applying transfer rules to the SL structure. While TL translations are constructed, beam search manages the large search space by ranking translation hypotheses and pruning the search by dropping lower scoring hypotheses. A number of stacks are used to organize translation hypotheses into groups of comparable hypotheses, according to the portion of SL structure that has already been translated to produce each hypothesis, i.e. hypothesis stack N stores TL translation hypotheses with N nodes covered in the SL structure. For example, Figure 5.3(a) shows the hypothesis stacks for decoding the deep syntactic structure of *Die Katze schläft gern* containing 4 nodes and therefore requiring stacks 1-4 for decoding.

Transfer rules are indexed by root node so that they can be retrieved quickly to translate SL structure nodes. For example, in Figure 5.3(a) the rules rooted at node *Katze* are stored together. Since rules are applied top-down to the SL structure (see Section 5.2) rules beginning at the root node of the SL structure are first used to construct hypotheses. For example, in Figure 5.3(b) the rule that translates the root node of the SL structure *schlafen* as *doze* is first used to construct a hypothesis and since it covers one SL node it is stored in hypothesis stack 1. Figure 5.3(c) shows the next three hypotheses that are constructed: *snooze*, *sleep* and *like sleep*. Hypotheses are ordered within each stack according to their score, high-to-low from bottom-to-top. We currently use histogram pruning. When a stack becomes full, lower scoring solutions are pruned by being popped off the top of the stack.

For efficiency, each partial translation is only stored once in memory even though it may be part of several different future hypotheses. For example, hypothesis stack 2 in Figure 5.3(d) contains four translations constructed by expanding hypothesis *doze* by four different rules, each translating the word *Katze* into a different TL

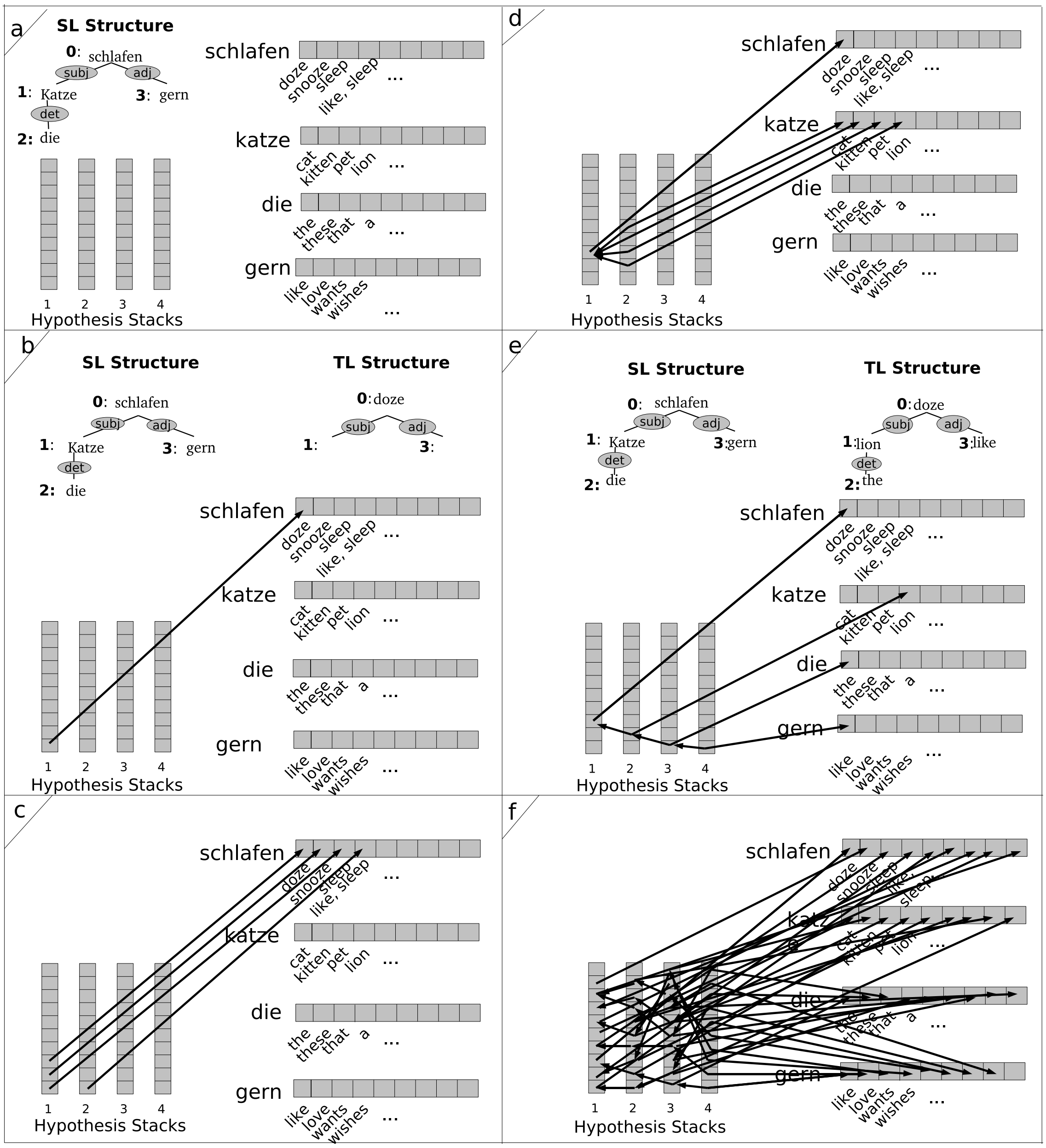


Figure 5.3: Beam Search Decoding of Example German Deep Syntactic Structure

word. These new hypotheses are represented by a reference to the most recently applied transfer rule (rules translating *Katze*) and a reference back to the previous hypothesis.

Figure 5.3(e) shows an example of how per single completed translation, the structure for *the lion likes to doze*, is represented in the hypothesis stacks and Figure 5.3(f) shows all hypotheses represented when the decoder has completed translating a single SL input structure. The m-best translated structures can be retrieved from the final stack. Later in Chapter 6, we investigate the effects on MT output of using different decoder beam sizes, in addition to generating from different size decoder output m-best lists.

5.4 Efficient Dependency-based Language Modeling

Although the search space is limited by beam search, during decoding large numbers of TL hypothesis structures need to be ranked. At each expansion of a translation hypothesis (via joining of an existing hypothesis with a transfer rule) a language model score for the newly created hypothesis needs to be computed. Since this is carried out very many times per single decoding run, it is vital that the method of computing this score is highly efficient.

In our system, we pre-compute a deep syntax language model score for each transfer rule prior to beam search. This score is computed only once for each rule even though a single rule may be part of several translation hypotheses. Then during decoding, when a translation hypothesis is expanded by adding a new rule, the new hypothesis score can be computed quickly by combining the score of the old hypothesis, the rule score and a score computed based on the probabilities of ngrams where the old hypothesis and rule join together. The probability of a TL hypothesis, h_n, produced by combining hypothesis h_{n-1} and rule r can be computed as follows:

$$hyp\_score(h_n) = hyp\_score(h_{n-1}) * join\_score(h_{n-1}, r) * rule\_score(r)$$

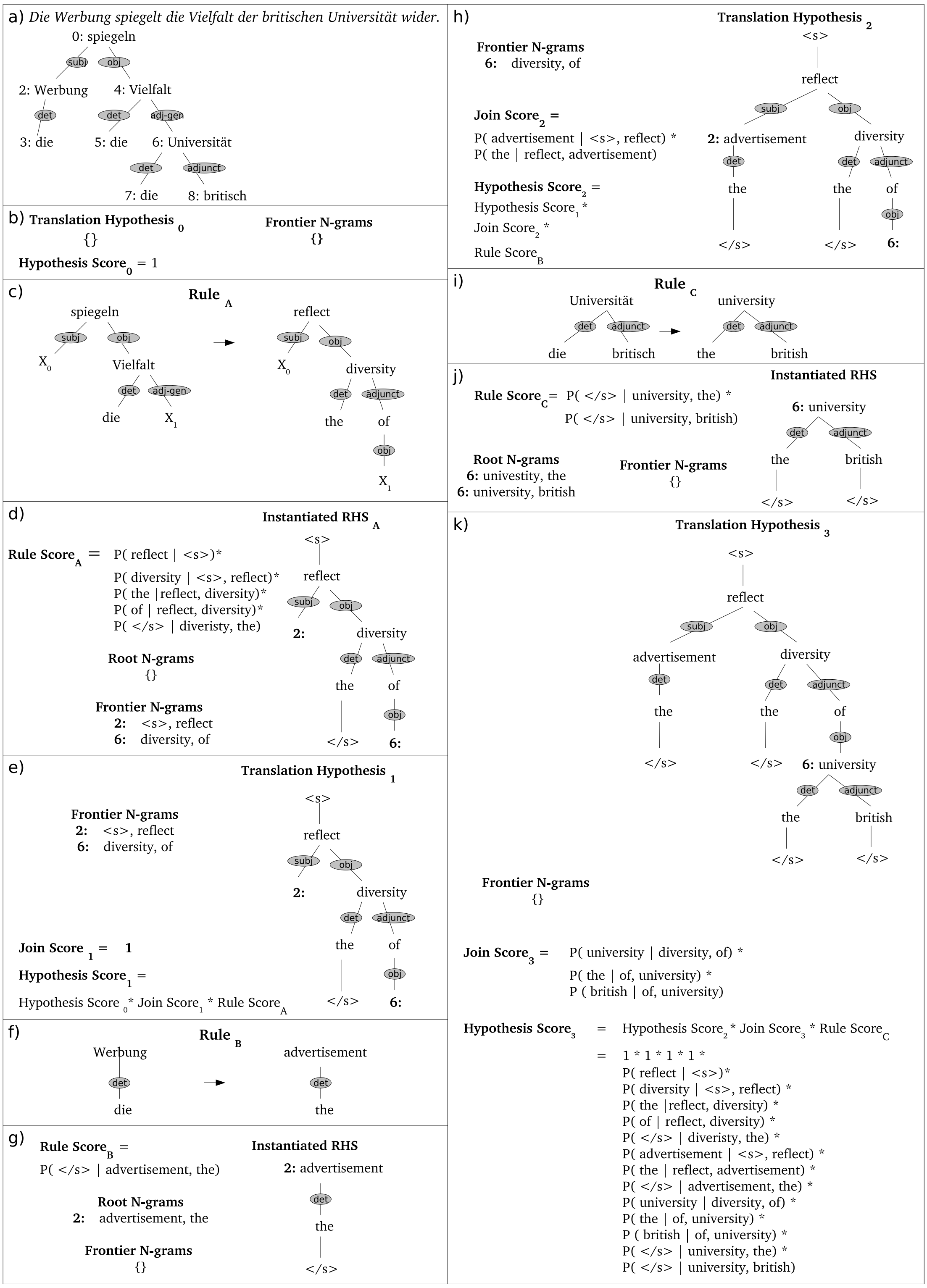


Figure 5.4: Efficient Deep Syntax Language Modeling

Since $hyp\_score(h_{n-1})$ and $rule\_score(r)$ are already computed, only $join\_score(h_{n-1}, r)$ needs to be computed to compute $hyp\_score(h_n)$.

Figure 5.4 shows how the language model scores are efficiently computed when decoding the deep syntactic structure for the German sentence *"Die Werbung spiegelt die Vielfalt der britischen Universität wider"* ("The advertisement reflects the diversity of the British university"). We begin with the German deep syntactic structure graph shown in Figure 5.4(a) with nodes labeled by id numbers. Figure 5.4(b) shows the initial empty translation hypothesis that has probability 1.

Figures 5.4(c), 5.4(f) and 5.4(i) show example transfer rules that can be applied to the German deep syntactic structure. Deep syntax language model scores are precomputed for each rule (by identifying all trigrams within the RHS structure and computing the product of their individual probabilities); we call this the $rule\_score$ (see Figure 5.4(d) for $Rule_A$, Figure 5.4(g) for $Rule_B$ and Figure 5.4(j) for $Rule_C$). In addition, for each rule, ngrams located at the RHS root node and frontier nodes are recorded. For example, $Rule_B$ in Figure 5.4(g) has a single root node bigram *advertisement-the* located at node 2, and $Rule_A$ in Figure 5.4(d) has two frontier bigrams *<s>-reflect* and *diversity-of* located at nodes 2 and 6, respectively. This information is used later when computing the score of joining a rule and a hypothesis.

Figure 5.4(e) shows the translation hypothesis established by applying $Rule_A$ to the German structure. The language model score for the structure is computed by combining the score of the previous hypothesis (since this is the first rule for this hypothesis, the previous hypothesis is the empty hypothesis and its score is therefore 1), the join score (since we are joining the rule with the empty hypothesis this score is also 1) and the rule score of $Rule_A$ (see Figure 5.4(d)).

Figure 5.4(h) shows the translation hypothesis created by expanding $Hypothesis_1$ by $Rule_B$. Since this expansion involved adding a rule at node 2 in the TL structure, the joining trigrams are derived by creating lists of words via all combinations of the frontier bigrams belonging to $Hypothesis_1$ labeled 2 and the root bigrams of $Rule_B$, also labeled 2 (see root ngrams in Figure 5.4(g)). For this example, this results in a single word sequence *<s>-reflect-advertisement-the* which forms two trigrams *<s>-reflect-advertisement* and *reflect-advertisement-the*. The score for $Hypothesis_2$ is then computed by combining the hypothesis score for $Hypothesis_1$, this join score and the precomputed rule score for $Rule_B$. Finally, $Rule_C$ is used to expand $Hypothesis_2$ to form the complete TL structure shown in Figure 5.4(k). Figure 5.4(k) again includes how we combine the previous hypothesis score, the join score and the rule score to compute the deep syntax language model score for $Hypothesis_3$.

5.5 Translating Atomic Features

Factored Models (Koehn and Hoang, 2007) can be used to incorporate different kinds of information into translation, such as CCG supertags, for example, aiding reordering of words in the TL language (Birch et al., 2007), and employ richer statistical translation models by translating lemmas separately from morphological (or morpho-syntactic) information. Factored Models also have the potential to increase coverage of unseen morphological inflections of words, since analysis and generation components can be trained on monolingual data.

We use an adaptation of Factored Models (Koehn and Hoang, 2007) for translating atomic features in our system. Using Factored Models for deep syntactic transfer requires some adaptation, however. When we compute the translation model we not only include lemmas but also dependency relations between lemmas, as described in Section 4.3.1. In addition, within our architecture generation must be carried out on the sentence level, as opposed to word level in Phrase-Based Factored Models. Due to sentence-level generation, the combinatorial explosion that can occur when combining TL lemmas and morpho-syntactic information in Factored Models is a much more severe problem for deep sytactic transfer, as a far higher proportion of translation options must be pruned. In the next section, we use the terms SL factor and TL factor, in place of SL atomic feature and TL atomic feature, simply to be

$$\begin{array}{ccc} \text{Mann} & \rightarrow & \text{man / gentleman / husband / worker / fellow} \\ \begin{bmatrix} \text{pers} & 3 \\ \text{num} & \text{singular} \\ \text{case} & \text{dative} \\ \text{gend} & \text{masculine} \\ \text{syn-n-type} & \text{common} \\ \text{common-n-type} & \text{count} \end{bmatrix} & & \begin{bmatrix} \text{num} & \text{singular / plural} \\ \text{pers} & \text{1 / 2 / 3} \\ \text{case} & \text{nominative / oblique} \\ \text{syn-n-type} & \text{common / pronoun / proper} \\ \text{common-n-type} & \text{count / gerund / mass / measure / partitive} \end{bmatrix} \end{array}$$

Figure 5.5: Example of the combinatorial explosion involved in translating lemmas and morpho-syntactic factors separately.

consistent with terminology in Koehn and Hoang (2007).

Using Factored Models allows us to use a statistically richer translation model for translating lemmas and dependency relations, in addition to richer models for translating morpho-syntactic information, and achieving coverage of inflections of lemmas not seen in bilingual training. The combinatorial of large numbers of translation options and our restriction to sentence-level generation results in severe pruning of translation options prior to generation. To combat the effects of this, we present *factor templates*, a method of limiting the number of morpho-syntactic factors that are translated separately from lemmas that also provides an effective way of translating idiosyncratic translations. Our method could potentially be used to improve idiosyncratic translations in general for Factored Models.

5.5.1 Combinatorial Explosion

The number of translation options for a SL word (or phrase) in Factored Models is $O(e^f)$, where f is the number of SL factors (including the lemma) and e is the number of possible translations for a SL factor. For example, Figure 5.5 shows the lemma and morpho-syntactic factors for the German word *Mann* and the possible translations into English. The total number of translation options in this simple example is 900. The task of guessing a single correct combination out of this large number of possible combinations can be challenging.

Sentence-level generation forces a large proportion of translation options to be pruned, and in fact in our system when we translate morpho-syntactic information separately from lemmas we only consider the single most probable translation for each SL factor.[2] Due to the large number of possible combinations of TL lemmas and morpho-syntactic information, achieving a good combination is challenging, and we use *factor templates* to help find good combinations.

5.5.2 Factor Templates

A factor template can be envisaged as a blue-print for translating a SL phrase into the TL. Each template has a source and target side containing the lemmatized words and an example set of morpho-syntactic factors for each source and target lemma. Figure 5.6(a) shows a factor template for the German-English phrase *neues Haus*|||*new house*.[3]

When translating a SL phrase, the target side of the factor template is used to provide an initial set of translated TL morpho-syntactic factors. The set of factors provided by the template may not contain the correct translation for all of the SL factors, and we use information in the source side of the template to indicate which factors may be incorrect. Figure 5.6(b) shows how the input SL phrase *neue Häuser* is decomposed into its lemmas, *neu* and *haus* and morpho-syntactic information. The SL factors of the input words are compared with the SL template factors. Only when a mismatch occurs between a template factor and an input factor, is a factor translated separately from its lemma. All target side factors in the template for which the corresponding SL factor matched that of the SL input are used as TL output factors. In the example in Figure 5.6(b), all SL factors in template 5.6(a) match those of the template except for *number*. Therefore, all TL template factors excluding *number* are used as the TL output factors for *new house*. The value of *number* can then be translated separately from the rest of the translation.

[2] We do generate the m-best TL structures output by the decoder, however.

[3] Morpho-syntactic factors in the example are obtained from Lexical Functional Grammar (LFG) f-structures.

(a) **Factor Template**

$$\begin{array}{c}\text{neu}\\ \begin{bmatrix}\text{degree} & \text{positive}\\ \text{a-type} & \text{attributive}\end{bmatrix}\end{array}\quad \begin{array}{c}\text{haus}\\ \begin{bmatrix}\text{pers} & 3\\ \text{num} & \text{sg}\\ \text{gend} & \text{neut}\\ \text{case} & \text{nom}\\ \text{syn...} & \text{common}\\ \text{com...} & \text{count}\end{bmatrix}\end{array}\quad \rightarrow \quad \begin{array}{c}\text{new}\\ \begin{bmatrix}\text{degree} & \text{positive}\\ \text{a-type} & \text{attributive}\end{bmatrix}\end{array}\quad \begin{array}{c}\text{house}\\ \begin{bmatrix}\text{pers} & 3\\ \text{num} & \text{sg}\\ \text{case} & \text{nom}\\ \text{syn...} & \text{common}\\ \text{com...} & \text{count}\end{bmatrix}\end{array}$$

(b) **Factored SL Input** **Factored TL Output**

$$\begin{array}{c}\text{neu}\\ \begin{bmatrix}\text{degree} & \text{positive}\\ \text{a-type} & \text{attributive}\end{bmatrix}\end{array}\quad \begin{array}{c}\text{haus}\\ \begin{bmatrix}\text{pers} & 3\\ \textbf{num} & \textbf{pl}\\ \text{gend} & \text{neut}\\ \text{case} & \text{nom}\\ \text{syn...} & \text{common}\\ \text{com...} & \text{count}\end{bmatrix}\end{array}\quad \rightarrow \quad \begin{array}{c}\text{new}\\ \begin{bmatrix}\text{degree} & \text{positive}\\ \text{a-type} & \text{attributive}\end{bmatrix}\end{array}\quad \begin{array}{c}\text{house}\\ \begin{bmatrix}\text{pers} & 3\\ \textbf{num} & \textbf{?}\\ \text{case} & \text{nom}\\ \text{syn...} & \text{common}\\ \text{com...} & \text{count}\end{bmatrix}\end{array}$$

Figure 5.6: **(a)** Factor template for German-English lemmatized phrase pair: *neu haus*|||*new house*, **(b)** Factored SL input phrase for *neue Häuser*, mismatching features in the source input are in bold and factors translated separately from the lemma have '?' as a value in the English factored phrase.

5.5.3 Extracting Factor Templates

Factor templates are automatically extracted from the annotated bitext corpus and only a single template is extracted for each distinct lemmatized transfer rule. For example, both of the following word sequences could exist in sentence pairs of the training data: **(1)** *neues haus ist* → *new house is*, **(2)** *neue häuser sind*→*new houses are*, and since the transfer rules extracted from both will contain the same lemmas and dependency relations, only a single factor template is extracted along with a single set of morpho-syntactic factors belonging to either one of the SMT phrases.

5.5.4 Avoiding the Combinatorial Explosion

Factor templates reduce the number of factors translated separately from each lemma and in doing so, reduce the overall number of translation options produced when combining translated lemmas and factors. For instance, in the example shown in Figure 5.6(b), the number of translation options considered for the English phrase with lemmas *new house* is 2 as opposed to the total possible 1620 (since *degree* has 3 possible values: *comparative, positive, superlative*; *a-type* 3 possible values: *adverbial, attributive, predicative*; *person* 3 values; *number* 2 values; *case* 2 values; *syn-n-type* 3 values and *common-n-type has 5 possible has values*).[4]

5.5.5 Idiosyncratic Translations

Factored Models can, in some cases, over-generalize and produce an incorrect translation that may not occur with non-factored Phrase-Based Models. *Valid* idiosyncratic translations exist for words in different language pairs and such exceptional translations can cause problems when the lemma is translated separately from its morpho-syntactic information. A classic example is when translating between two languages in which a noun with the same meaning has a different *number* in each

[4] Note that there is an even larger total number of translation options for *neue Häuser*. In the example, we just include translation options for English lemmas *new house*

language.[5] For example, consider the German phrase *"die Polizei ist"* in which the noun *Polizei* is in the singular and the correct English translation is *"the police are"* in which the translation of *Polizei* is the noun *police* which must be in the plural in English. For Factored Models, if the morpho-syntactic factor *number=singular* of the German lemma *Polizei* is translated into English separately from the lemma *Polizei*, there is a risk of over-generalizing and assigning a high probability to *police, number=singular* in English, which is incorrect. For deep syntactic transfer Factored Models, this over-generalization problem is severe, since generation operates on the sentence level, compared to the word level in Phrase-Based Factored Models. Since so many translation options are pruned, it's very unlikely for a deep syntactic transfer Factored Model to produce the correct translation, *the police, number=plural* if *number* is translated separately from *Polizei.*

Factor templates provide a solution to over-generalizing when translating lemmas separately from their morpho-syntactic information. Figure 5.7(a) shows a factor template and Figure 5.7(b) shows how the template is applied to an input German structure. Since only the factor *tense* mismatches the source side of the template, it is the only factor to be translated separately from the lemma and all other factors, including idiosyncratic *number*, are provided by the target side of the factor template. This results in the idiosyncratic translation of *Die Polizei kommt* (where *polizei, number=singular*) being translated correctly into English as *The police are coming* (where *police, number=plural*).

5.5.6 Translating Mismatching Factors

As described in Section 5.5.2, factors in the SL input that mismatch those of the factor template are translated separately from the lemma. For translating mismatching factors in the word-aligned annotated corpus, we use a probability distribution computed from the relative frequencies of source and target factors, $p(v_e|v_f)$, where v_f

[5] The following example is borrowed from Philipp Koehn's Factored Models tutorial at the Third Machine Translation Marathon, January 2009.

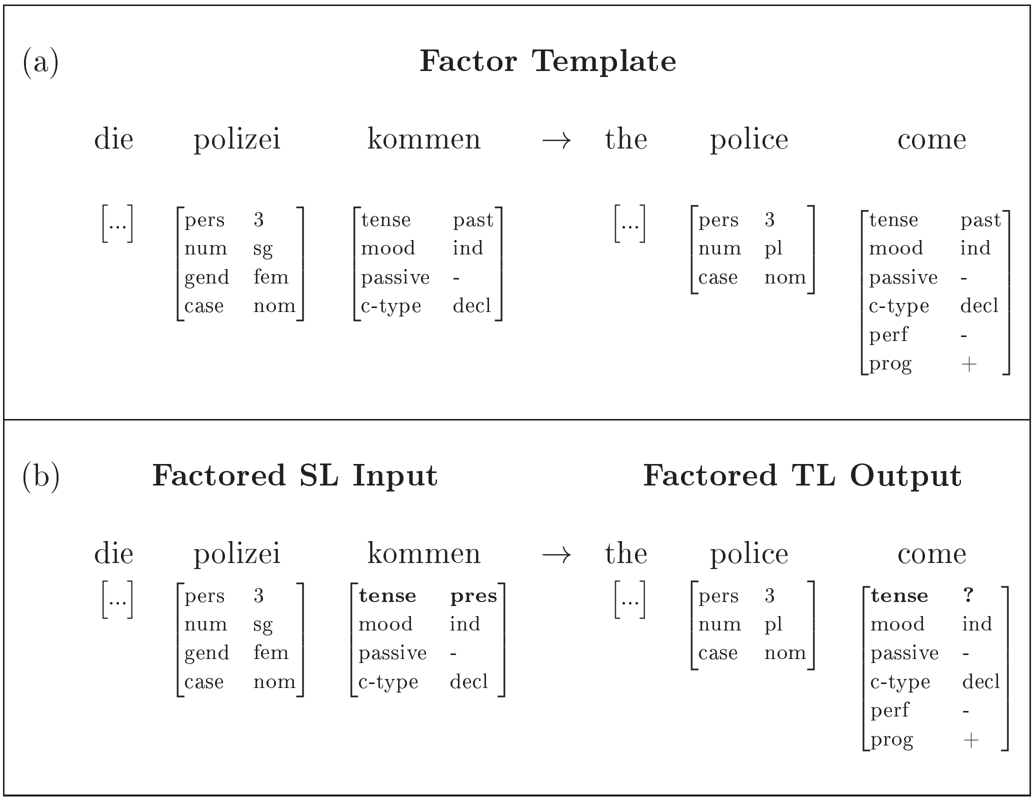


Figure 5.7: Factor templates correctly handle exceptions to the rule: **(a)** factor template for the German-English lemmatized phrase pair: *die polizei kommen*|||*the police come*, **(b)** factored SL input phrase *Die Polizei kommt* correctly translates *Polizei* from singular in German into plural in English, mismatching features in the source input are in bold and factors translated separately from the lemma have value '?'

denotes a SL factor and v_e is a target language factor.

In addition, since the dependency relation between a TL word and its head is also available in TL structures, this may be useful for translating the morpho-syntactic factor *case*.[6] Therefore, we also compute relative frequencies for *case* conditioning on the dependency relation between a word and its head, $p(v_e|d_e)$, where d_e denotes the dependency relation between a TL word and its head. Later in Section 6.2.4, we investigate the effects on MT output of using factor templates as well as using these alternate probability distributions for translating morpho-syntactic factors.

5.6 Conclusion

In this chapter, we presented how SL deep syntactic structures are decoded to produce TL deep syntactic structures by applying transfer rules top-down to the SL structure. Transfer rules are used to transfer snippets of SL structure to the TL with variables providing information on the appropriate location of each translated snippet in the TL structure. We described how we carry out a heuristic search for the m-best TL deep syntactic structures and how we efficiently incorporate a deep syntax language model. Finally, we described how we use an adaptation of Factored Models and factor templates to translate SL atomic features to the TL. In the next chapter, we provide an experimental evaluation of the system using different resources and variations of the methods described so far.

[6] Suggested by Philipp Koehn, December 2009.

Chapter 6

Evaluation

6.1 Introduction

In this chapter, we include an experimental evaluation of the system and its components. In addition, we include an evaluation of the deep syntax language model in which we compare ngram coverage of our model to that of a standard string-based language model on a held-out test set.

6.2 SMT via Deep Syntactic Transfer Experiments

We provide a detailed evaluation of the system to investigate effects on MT performance of using (i) different methods of word alignment, (ii) different methods of translating atomic features, (iii) restricting the size of transfer rules used to translate SL structures to the TL,[1] (iv) different beam sizes during decoding, (v) generating different sized m-best TL decoder output structure lists, and (vi) different k-options for deterministic versus non-deterministic generation. In addition, we train a state-of-the-art PB-SMT system, Moses (Koehn et al., 2007), with the same training data to evaluate how far off state-of-the-art performance the system currently is, and also to investigate if our deep syntax SMT system produces the same kinds of translations as a PB-SMT system, examining the translation of one syntactic construction in particular, the compound noun. We investigate for this particular syntactic construction, if our system achieves state-of-the-art performance by providing a human evaluation of the translation of the first 100 German compound nouns in the test data.

6.2.1 Training

German and English Europarl (Koehn, 2005) and Newswire sentences length 5-15 words were used as bilingual training data, and were parsed with XLE (Kaplan et al., 2002) and LFG Grammars (Kaplan et al., 2004; Riezler et al., 2002), resulting in

[1] For example, if the limit is 2, only rules with a maximum of 2 nodes in the LHS and a maximum of 2 nodes in the RHS are used for transfer.

approximately 360K sentences pairs and the single best parse for both source and target was selected.[2] A deep syntax language model was trained on the LFG-parsed English side of the Europarl corpus, approximately 1.26M English f-structures, again using only the single-best parse, and a trigram deep syntax language model was computed by extracting all unigram, bigram and trigrams from the f-structures before running SRILM (Stolcke, 2002). The surface form language model, used after generation, consisted of the English side of the Europarl and again was computed using SRILM.

Minimum Error Rate Training (MERT) (Och, 2003) was carried out on 1000 development sentences using Zmert (Zaidan, 2009), open source tool, maximising for BLEU (Papineni et al., 2001, 2002). MERT was run separately for each word alignment method with the following settings: rule size limit = none, beam = 20, m = 100, k-option = shortest, and due to amount of computation time needed for running MERT for the system, these weights were used for each additional experiment.

6.2.2 Evaluation Metrics

Since MT system coverage changes with each configuration, before running automatic evaluation metrics, empty strings produced by the system for sentences that are out of coverage are replaced by their SL input sentence. In addition to standard MT evaluation metrics, such as BLEU (Papineni et al., 2001, 2002) which we use as our main standard automatic evalution metric, since it is the most widely used metric in SMT, we use a method of evaluation adopted from LFG parser evaluation that compares parser-produced f-structures against gold-standard f-structures. The method extracts triples that encode dependency relations, such as *subject(enhance,proposal)* and *object(enhance,safety)* for sentence *The proposal will enhance the safety of feed* for example, and triples encoding morpho-syntactic information, for example *case(proposal,nominative)* or *tense(enhance,future)*, from each parser produced f-structure and corresponding gold-standard f-structure, counting

[2] The same parsing resources used in Riezler and Maxwell (2006).

matching triples to finally compute a single precision, recall and f-score computed over the triples of the entire test set. We evaluate the *highest ranking TL decoder output f-structure* with an adaptation of this method since we do not have access to gold-standard f-structures for the test set. Instead we use the next best thing, the *parsed reference translations* (similar to Owczarzak (2008); Owczarzak et al. (2008, 2007b,c,a)). This provides an evaluation that eliminates generator performance and gives a breakdown of results for individual dependency relations and atomic features. Note, however, that this method of evaluation is somewhat harsh when used for the purpose of MT evaluation. Since it was designed to evaluate parser output, it assumes correct lexical choice, so, for example, if the MT system produces the correct tense but a different lexical item for *enhance*, such as *tense(improve,future)*, the triple is counted as incorrect, ignoring the fact that tense was in fact correct. Correct triples, in the evaluation, are those where the correct lexical choice was made by the system *and* the correct dependency relation (or morpho-syntactic information) was produced.

6.2.3 Experiment: Word Alignment

An alignment between the nodes of the SL and TL deep syntactic training structures is required in order to automatically extract transfer rules. In our evaluation, we investigate the following three methods of word (or node) alignment, all using Giza++ (Och et al., 1999) for alignment and Moses (Koehn et al., 2007) for symmetrization:

- SF-GDF: input the surface form bitext corpus to Giza++ and symmetrize with grow-diag-final yielding many-to-many alignments between surface form words. Then map this alignment from each word to its corresponding word in the deep syntactic structure. This yields up to a many-to-many alignment between deep syntactic structure nodes and was used in Riezler and Maxwell (2006).[3]

[3] It should be noted that we use an entirely different method of transfer rule extraction in our

Word Align.	Align. Pts.		Rules	
	Total	Ave.	Total	Ave.
SF-GDF	4.5M	12.5	2.9M	8.1
DS-GDF	4.1M	11.5	9.7M	27.1
DS-INT	2.5M	6.9	13.9M	38.8

Table 6.1: Statistics on number of transfer rules extracted for different word alignment methods

- DS-INT (our main method of word alignment described in Section 3.3.1): reconstruct a bitext corpus by extracting lemmas from the deep syntactic training structures, input the reconstructed bitext to Giza++, and use the intersection of the bidirectional word alignment for symmetrization. This yields a one-to-one alignment between deep syntactic structure nodes.

- DS-GDF: reconstruct a bitext corpus by extracting lemmas from deep syntactic training structures and input the reconstructed bitext to Giza++ (as in DS-INT), but use grow-diag-final for symmetrization yielding up to many-to-many alignments between deep syntactic structure nodes.

Each method of word alignment was run on the training data yielding an alignment between local f-structures within each training f-structure pair. All transfer rules consistent with this alignment were extracted.

Results

Table 6.1 shows statistics for each word alignment method and Table 6.2 shows automatic evaluation results. DS-INT by far achieves the best result with a BLEU score of 16.18%. Results drop sharply when the grow-diag-final algorithm is applied to deep syntax word alignment (DS-GDF), with scores of 6.04% BLEU. The method of word alignment that uses the surface form bitext corpus for word alignment (SF-GDF) achieves an extremely low score of only 1.61% BLEU.

evaluation, we do not correct word alignment and do not include hand-crafted transfer rules.

Word Align.	BLEU	Precision	Recall	F-score	Prec. Gram. Coverage
SF-GDF	1.61 %	15.83 %	5.46 %	8.12 %	1.77 %
DS-GDF	6.04 %	29.13 %	28.17 %	28.64 %	7.98 %
DS-INT	16.18 %	40.31 %	41.25 %	40.78 %	38.01 %

Table 6.2: Effects of using different methods of word alignment. Note: rule size limit = none, beam = 100, m = 100, k = 1, k-option = shortest

Discussion

Experiment results show that the performance of the system can vary quite a bit depending on how word alignment is carried out and this is caused by each method of word alignment constraining transfer rule extraction differently. In general, the *more alignment points* for a pair of f-structures, the *fewer transfer rules* extracted. Table 6.2 shows DS-INT yields fewest alignment points (6.9 per sentence pair (psp)) and subsequently most consistent transfer rules (38.8 psp), while the extension of this method that uses grow-diag-final (DS-GDF) yields more alignment points (11.5 psp) with fewer consistent transfer rules (27.1 psp).

It is not only the number of alignment points that effects the number of consistent transfer rules, but also the *level of isomorphism between alignment points within pairs of f-structures* in the training corpus. The *less isomorphic* a pair of f-structures is with respect to the position of aligned nodes, the *fewer consistent rules.* The effects of this can be seen when we compare the number of consistent rules produced by DS-GDF and SF-GDF, which have a relatively similar number of alignment points, 11.5 psp and 12.5 psp respectively, but yield very different numbers of consistent transfer rules, 27.1 psp and 8.1 psp respectively. Carrying out word alignment on the surface form sentences, as opposed to deep syntactic structures, yields a much less isomorphic alignment between local f-structures and subsequently far fewer consistent transfer rules.

In addition, the fact that each method yields different *quality* alignment points should be taken into account. The method of alignment based on the surface form sentences (SF-GDF) yields a lower quality alignment since alignment is run on a more specific representation, the surface form words as opposed to lemmas in both the deep syntax methods. This is observed in the precision and recall results for the comparison of the MT system produced f-structures and parsed reference translations (Table 6.2), as the SF-GDF, although yielding far fewer transfer rules, yields lower quality transfer rules, since its precision is lower, 15.83%, than both precision scores for methods of word alignment run on the lemmatized training data, 29.13% for DS-GDF and 40.31% for DS-INT. Since what we ultimately need is an alignment between local f-structures and not surface form words, alignment methods that work via the surface form sentences unnecessarily increase data sparseness by using the more specific surface form of words instead of lemmas of local f-structures resulting in lower quality alignment.

Symmetrization also effects the quality of alignment points. DS-INT yields a more reliable set of alignment points than DS-GDF, since DS-INT only contains alignment points found when word alignment is run in both language directions. For DS-GDF, the combination of adding some low quality alignment points and increasing the overall number of alignment points (and thereby over-constraining rule extraction) results in lower quality translations.

6.2.4 Experiment: Translating Atomic Features/Morpho-syntactic Factors

In this experiment, we investigate the different methods of translating atomic features described in Section 5.5. We try five different configurations, keeping all other resources used for training and testing constant: (i) plain factored: all atomic features are translated separately from lemmas using $p(v_e|v_f)$, (ii) factored + *case* special: all atomic features except *case* are translated separately from lemmas using $p(v_e|v_f)$ and *case* is translated separately from the lemma using $p(v_e|d_e)$, (iii) plain templates: the target side atomic features in the template of each phrase is used as-is with no factoring, effectively disregarding SL input atomic features, (iv) templates + mismatching factored: templates are used to translate matching atomic features and mismatching atomic features are translated using $p(v_e|v_f)$, (v) templates + mismatching factored + *case* special: templates are used to translate matching atomic features and all mismatching atomic features except for *case* are translated using $p(v_e|v_f)$ and *case* is translated using $p(v_e|d_e)$.

Atomic Feature Translation

For atomic feature translation, relative frequencies for corresponding atomic features were computed from the word-aligned training corpus, and we include the probability distribution of a selection of atomic features in Table 6.3. For configurations (ii) and (iv), relative frequencies of *case* given the dependency relation of a word with its head were computed from the parsed TL side of the bitext corpus, and a selection of the probability distributions are shown in Table 6.4.

Results

Table 6.5 shows BLEU scores for the MT system for each method of translating factors.[4] The results show a low baseline for the plain factored model, in which all factors are translated separately from lemmas, with a BLEU score of 6.23%. Using the dependency relation between a word and its head to translate *case* increases the results slightly to 6.27% BLEU. When target side factors are taken directly from the factor templates, with no factors translated separately, this results in an improvement, increasing the BLEU score to 8.8%. The two methods that use factor templates to translate all matching SL factors perform best, improving the BLEU score substantially to 16.18% when all factors are translated with $p(v_e|v_f)$, with an additional improvement seen when the probability is conditioned on the dependency relation, $p(v_e|d_e)$, for translating *case*, increasing to 16.85% BLEU.

Table 6.5 also shows precision, recall and f-score results of translated triples

[4] BLEU+tc scores are BLEU with true casing.

Morpho-syntactic Factor	v_f	v_e	$p(v_e\|v_f)$
PERSON	1	1	0.97
		3	0.02
		2	0.01
	2	2	0.66
		3	0.30
		1	0.04
	3	3	0.97
		2	0.02
		1	0.01
TENSE	past	present	0.61
		past	0.38
		future	0.01
	present	present	0.88
		past	0.06
		future	0.06
NUMBER	singular	singular	0.94
		plural	0.06
	plural	plural	0.86
		singular	0.14
CASE	nominative	nominative	0.85
		oblique	0.15
	accusative	oblique	0.89
		nominative	0.11
	dative	oblique	0.88
		nominative	0.12
	genitive	oblique	0.91
		nominative	0.09
PASSIVE	-	-	0.96
		+	0.04
	+	+	0.74
		-	0.26
MOOD	indicative	indicative	0.99
	subjunctive	indicative	0.91
		subjunctive	0.08
	imperative	indicative	0.58
		imperative	0.42

Table 6.3: $p(v_e|v_f)$ for a selection of atomic features computed from 360K node-aligned German-English LFG f-structure pairs, probabilities are rounded to 2 decimal places, feature values with $p(v_e|v_f) < 0.01$ are omitted

Dependency Relation (d_e)	Case (v_e)	$P(v_e\|d_e)$
MODIFIER	obl	1.00
OBJECT	obl	1.00
THETA OBJECT	obl	1.00
OBLIQUE AGENT	obl	1.00
OBLIQUE	obl	1.00
OBLIQUE PARTICLE	obl	1.00
INTEROGATIVE PRONOUN	obl	1.00
TOPIC	obl	1.00
RELATIVISED TOPIC	obl	1.00
SUBJECT	nom	0.99
	obl	0.01
INTEROGATIVE FOCUS	obl	0.98
	nom	0.02
RELATIVE PRONOUN	obl	0.97
	nom	0.03
FRAGMENT	nom	0.96
	obl	0.04
X-COMPLEMENT	nom	0.81
	obl	0.19

Table 6.4: $p(v_e|d_e)$ computed from 360K English LFG f-structures, probabilities are rounded to 2 decimal places, feature values with $p(v_e|d_e) < 0.01$ are omitted

Factor Translation	BLEU	BLEU+tc	Prec.	Recall	F-score	Prec. Gram. Coverage
plain factored	6.23 %	5.66 %	34 %	32 %	33 %	12.14 %
factored + *case* special	6.27 %	5.70 %	34 %	32 %	33 %	13.39 %
plain templates	8.80 %	8.09 %	35 %	33 %	34 %	20.34 %
templates + mismatching factored	16.18 %	15.20 %	40 %	41 %	41 %	38.01 %
templates + mismatching factored + *case* special	16.85 %	15.79 %	40 %	41 %	41 %	41.20 %

Table 6.5: Automatic evaluation results on 1755 held-out German-English sentence pairs

when compared to those of the parsed reference translations. The results are in line with the BLEU scores of Table 6.5, with respect to the rank of each method. For the Factored Models with and without using templates, when we condition the probability used to translate the morpho-syntactic factors on the dependency relation as opposed to the SL factor for *case*, we see no increase in f-score, as the f-score for both configurations without templates is 33% and for both configurations with templates it is 41%. The improvement from the baseline plain factored model when compared with the mismatching factor template methods is substantial, from an f-score of 33% to 41%, an increase of 8 percentage points absolute.

Table 6.6 shows a break-down of translation results for individual morpho-syntactic factors when compared to those of parsed reference translations. The best result for translating each morpho-syntactic factor is achieved using factor templates to translate matching factors only translating mismatching factors separately from the lemma. Although, BLEU scores improve when *case* is translated using probability conditioned on the dependency relation (16.85) as opposed to SL factor (16.18), between a word and its head, rather surprisingly we do not observe the same effect in the f-scores for *case*, as it remains at 46% for both configurations.

Morph-syntactic Factor	Factored Translation Method	Precision	Recall	F-score
CASE	factored	40 %	39 %	40 %
	factored + *case* special	41 %	40 %	40 %
	template	38 %	36 %	37 %
	template + mismatching factored	45 %	48 %	46 %
	template + mismatching factored + *case* special	45 %	47 %	46 %
PERSON	factored	49 %	42 %	45 %
	template	50 %	43 %	46 %
	template + mismatching factored	54 %	54 %	54 %
TENSE	factored	38 %	31 %	34 %
	template	37 %	30 %	33 %
	template + mismatching factored	39 %	33 %	36 %
NUMBER	factored	48 %	41 %	44 %
	template	46 %	39 %	42 %
	template + mismatching factored	53 %	53 %	53 %
PASSIVE	factored	39 %	31 %	34 %
	template	40 %	32 %	35 %
	template + mismatching factored	42 %	36 %	39 %
MOOD	factored	42 %	34 %	38 %
	template	43 %	34 %	38 %
	template + mismatching factored	45 %	38 %	41 %

Table 6.6: Results of comparison of a selection of automatically translated morpho-syntactic factors and reference translation morpho-syntactic factors

Discussion

Results show that using factor templates to decide which factors to translate separately from lemmas improves machine translation output significantly for our system. Accurately translating all factors separately from lemmas is difficult due to the very large number of possible combinations of values and the fact that generation in the MT system is carried out on the sentence level. Since only 100 TL structures (m=100) are generated per TL sentence, a very high proportion of translation options are pruned and using factor templates to translate factors that match the source input factors results in the pruning of a lower number of high quality translation options and subsequently considerably higher BLEU, precision and recall scores.

Conditioning probabilities for translating *case* using the TL dependency relation of a word improves BLEU score and this is caused by the relatively large increase this causes in the number of translations that now fall within coverage of the precision grammar used for generation, as it increases from 38.10% to 41.2%, an increase of over 3%, showing that using the TL dependency relation of a word for translating *case* is definitely worthwhile. As mentioned earlier, the improvement is not reflected in the fscore for *case* as recall in fact decreases (by 1%) when the dependency relation is used for translating *case* and this is probably caused by the fact that the dependency relations themselves are decided automatically by the MT system, so when we compare them to the parsed reference fewer are correct because the dependency relation in the parsed reference translation is also different. The decrease in recall should not be interpreted as a negative, as the important thing is not to match the *case* of each word to that of the parsed reference translation, but to produce the correct *case* in each individual TL structure, as this causes fewer generation clashes.

Examining the probability distributions computed from the word-aligned corpus for translating individual morpho-syntactic factors reveals some interesting insights into how factors correspond between the German and English words of the corpus

(Table 6.3).[5] The probability distribution for *person* shows that in the training data, only approximately 66% of nouns in the 2^{nd} person in German are translated into English as the 2^{nd} person, with 30% being translated as 3^{rd} person and 4% being translated as 1^{st} person. Another surprising statistic is observed in the probability distribution for *tense*, that 61% of verbs in the *past* tense are translated into the *present* tense in English, with a smaller portion translated as *past*, 38%, and a minimal amount as *future* (1%). However, it's worth mentioning that *tense* at the f-structure level of analysis in LFG is not simply divided notionally into *past*, *present* and *future*. For example, the tense of the German verb *gehen* in *Ich ging* is analyzed as:

- *tense past*, mood indicative,

and the corresponding verb in its English translation *I went* is given a similar analysis for tense:

- *tense past*, progressive -, perfect -, mood indicative,

but the alternate translation *I have gone* is analyzed as follows

- *tense present*, progressive -, *perfect +*, mood indicative.

So, although notionally both English translations encode that the event was in the past, syntactically only the former is in the *past* tense, and this phenomenon probably accounts for much of the divergence in *tense* observed in the probability distribution.

In addition, the probability distribution for *number* in Table 6.3 shows that a relatively large proportion of nouns that appear in the plural in German are translated into a singular noun in English, 14%. It's not surprising that the values for *case* between German and English do not correspond well and even when a German noun is in the nominative, only 85% of the time is it translated into the

[5] Note that these statistics are computed from the *automatically* aligned bitext corpus, so the margin of error introduced by the (lower than gold-standard quality) alignment must be taken into account.

nominative case in English. When translating the *case* of a noun, the dependency relation between a word and its head is more informative than the source language *case* factor, as can be seen from Table 6.4, although since we construct the TL structure automatically via the translation of the SL structure, we need to take into account that TL dependency relations themselves may be incorrect, but as we mentioned earlier if high generator precision grammar coverage is our priority a good combination of TL dependency relation and case is more important than achieving the case of the word in the reference translation.

6.2.5 Experiment: Limiting Transfer Rule Size

In this experiment, we investigate imposing a limit on the size of transfer rules used for transfer by the MT system. Transfer rules were filtered by the maximum number of nodes/words per LHS and RHS ranging from a limit of a maximum of 1 node per LHS and RHS to a maximum of 7 nodes.

Results

Table 6.7 shows the automatic evaluation results for the MT system for each rule size limit. As the limit on the size of transfer rules increases from a limit of 1 node to a limit of 7, so does the BLEU score, from 10.09% to 16.55%, with a slight decrease when no limit is put on the size of transfer rules. The biggest increase is seen when we compare the results when the limit is increased from 1 node (10.09% BLEU) to 2 nodes (14.94% BLEU), an increase of over almost 5 percentage points. Precision, recall and f-score in general increase as we increase the limit on rule size, for example, from an f-score of 36.12% when the limit is 1 to 40.74% for a limit of 7.

Discussion

In general as we include large transfer rules, results improve due to larger parts of the SL deep syntactic structure being translated together, resulting in the TL structure

Max Rule Size	BLEU	BLEU+tc	Precision	Recall	F-score
1	10.09 %	9.30 %	38.67 %	33.89 %	36.12 %
2	14.94 %	13.89 %	41.55 %	39.09 %	40.28 %
3	15.85 %	14.83 %	41.50 %	39.93 %	40.70 %
4	16.31 %	15.26 %	41.03 %	40.25 %	40.63 %
5	16.14 %	15.15 %	40.75 %	40.50 %	40.62 %
6	15.52 %	14.62 %	40.31 %	40.71 %	40.51 %
7	16.55 %	15.51 %	40.46 %	41.03 %	40.74 %
none	16.18 %	15.20 %	40.31 %	41.25 %	40.78 %

Table 6.7: Effects of limiting transfer rule size. Note: word alignment = DS-INT, beam = 100, m = 100, k = 1, k-option = shortest

being constructed from large TL snippets of structure, which already contain fluent combinations of words. In addition, larger snippets of TL structure are more likely to be grammatical and result in successful generation. The minor decrease observed when we change from a limit of 7 to no limit on transfer rule size is probably due to a small number of erroneous transfer rules being eliminated when rule size is limited.

6.2.6 Experiment: Transfer Decoder Beam Size

In this experiment, we investigate the effects on MT output when the beam size of the decoder is increased to different sizes. In theory, increasing the beam size could increase the quality of MT output, as a higher number of possible solutions are reached by the search.

Results

Results for each beam size are shown in Table 6.8. Automatic evaluation results show that changing the beam size does not have a dramatic effect on system performance. For all tested beam sizes, 1-400, the BLEU score is around 13% with small variations. The f-score doesn't change dramatically either as it is approximately 41% for all beam sizes.

Beam Size	BLEU	BLEU+tc	Precision	Recall	F-score
1	12.76 %	11.78 %	40.61 %	41.19 %	40.90 %
5	12.84 %	11.82 %	40.70 %	41.54 %	41.11 %
10	13.03 %	11.97 %	40.79 %	41.43 %	41.11 %
20	12.83 %	11.76 %	40.69 %	41.31 %	41.00 %
50	12.69 %	11.66 %	40.35 %	41.18 %	41.00 %
100	12.67 %	11.65 %	40.31 %	41.25 %	40.78 %
200	12.67 %	11.62 %	40.24 %	40.99 %	40.61 %
400	12.52 %	11.50 %	40.06 %	40.78 %	40.78 %

Table 6.8: Effects of increasing the decoder beam size. Note: word alignment = DS-INT, rule size limit = none, m = 1, k = 1, k-option = shortest

Discussion

Increasing the beam size of the heuristic search does not increase the MT system performance. This indicates the possibility that the optimization using MERT and BLEU is not effectively optimizing the weights used during transfer decoding. A possible way to improve transfer decoder weight optimization would be to use an evaluation metric that operates directly on decoder output, as opposed to BLEU, which is applied to surface-form sentences generated from the deep syntactic structures. One such method is to use the f-score evaluation metric for MERT training, which may provide a better set of weights for transfer decoding. Due to time constraints, we leave this investigation to future work, however.

6.2.7 Experiment: Generating from m-best Decoder TL Output Structures

In this experiment, we investigate the effects on MT output quality of generating from different size decoder output lists. Increasing the number of TL decoder output structures that are generated from, decreases the number of translation options that are pruned prior to generation, and this reduces the likelihood of eliminating good translations at this stage in the pipeline.

m-best list size	BLEU	BLEU+tc
1	12.67 %	11.65 %
10	15.24 %	14.17 %
100	16.18 %	15.20 %
1000	16.57 %	15.52 %

Table 6.9: Effect of increasing the size of the m-best decoder output lists. Note: word alignment = DS-INT, rule size limit = none, beam = 100, k = 1, k-option = shortest. Precision = 40.31%, recall = 41.25%, f-score = 40.78%

Results

Table 6.9 shows automatic evaluation results for different m-best list sizes.[6] Results show that increasing the size of the m-best list of TL structures produced by the decoder, has quite a dramatic effect on system performance, with the largest increase in results observed when we increase the size of m from 1 (12.67% BLEU) to 10 (15.34% BLEU), an increase of almost 3 BLEU points absolute. Results increase again when we increase m to 100 (16.18% BLEU) and again to 1000 (16.57%). We include BLEU scores for true casing, and, as expected, for all configurations the BLEU score is lower (by approximately 1 BLEU point absolute in each configuration).

Discussion

Increasing the number of structures generated (Table 6.9) has a relatively dramatic effect on the quality of MT output. When m is increased from 1 to 10, an increase of almost 3 BLEU points absolute is observed and scores increase again when we move to 100 structures by almost 1 BLEU point. Increasing the size of m to 1000 results in an additional increase of 0.39 BLEU points absolute, but a relatively severe trade-off exists as the increase in computation time required for generation by increasing m from 100 to 1000 is significant, from approximately 2.33 to 26.75 cpu minutes per test sentence.

[6] Precision, recall and f-scores are the same for each configuration, since scores are computed on the highest ranking TL structure, which is the same in each configuration.

6.2.8 Experiment: Deterministic vs. Non-deterministic Generation

In this experiment, we investigate the effect on MT output quality of using deterministic versus non-deterministic generation. The deterministic k-options possible with XLE, *shortest* and *longest*, might in some cases select the best output for an input TL structure, but will inevitably cause some good translations to be pruned as the generator is forced to select a single output translation for each input TL structure. Using non-deterministic generation, allows all possible strings to be generated for each input structure, eliminating the possibility of the generator pruning good translations at this stage in the pipeline.[7]

Results

Table 6.10 shows automatic evaluation results for deterministic versus non-deterministic generation.[8] The lowest result is seen for deterministic generation with k-option *longest* (15.55%), where the generator outputs the longest result, while selecting the shortest generator output string for each TL structure results in an increase to 16.18% BLEU, by almost 1 BLEU point. When non-deterministic generation is used and the generator produces all TL strings for the TL input structure the score increases again to 17.29% BLEU.

Discussion

Allowing non-deterministic generation (Table 6.10) results in a significant increase in BLEU score. With respect to the trade-off in additional computation time required by non-deterministic generation, non-deterministic generation indeed is worthwhile,

[7] It's important to remember that the size of the n-best list of translations that is ultimately generated is $m*k$, where m is the size of the decoder output list and k is the number of structures generated from each TL structure (the value of k is likely to change from one structure to the next), so deterministic generation reduces the size on the n-best list of TL translations to m and not 1, which would be an easy mistake.

[8] Precision, recall and f-scores are the same for each method, since these scores are computed on the highest ranking TL structure before generation is carried out.

k-option list size	BLEU	BLEU+tc
longest	15.55 %	14.54 %
shortest	16.18 %	15.20 %
allstrings	17.29 %	16.13 %

Table 6.10: Deterministic versus non-deterministic generation. Note: word alignment = DS-INT, rule size limit = none, beam = 100, m = 100. Precision = 40.31%, recall = 41.25% and f-score = 40.78% for three configurations.

since the average time for generation is only increased by half a cpu minute per test sentence, from 2.33 (shortest) to 2.83 (allstrings) cpu minutes.

6.2.9 Experiment: Comparison with State-of-the-Art

In this experiment, we compare the performance of a state-of-the-art PB-SMT system, Moses (Koehn et al., 2007), with our deep syntax system. In our investigation, we examine if our system produces the same kinds of translations as the Phrase-Based system, focusing on one specific syntactic construction, the German Compound Noun (GCN), to observe if, for this particular syntactic construction, our system can achieve state-of-the-art performance in a human evaluation of the first 100 GCNs in the test data. A single human evaluator was used, who was presented with the correct English translation of the noun and the two system outputs in a blind test.[9] The same data as in previous experiments was used for training and testing of both systems. The deep syntax configuration settings were as follows: word alignment = DS-INT, rule size limit = none, beam = 100, m = 100, k-option = allstrings.

Results

Table 6.11 contains automatic evaluation results for the Deep Syntax (DS) system (17.29% BLEU) compared to the Phrase-Based (PB) system (30.7% BLEU) showing the degree to which our system currently under-performs compared to state-of-the-

[9] Due to lack of resources, the author acted as human evaluator.

	BLEU	Correct GCNs	Fuzzy GCNs	Precision Grammar Coverage
DS	17.29 %	56 %	25 %	38%
PB	30.70 %	54 %	22 %	n/a

Table 6.11: Comparison with state-of-the-art

	BLEU	HBLEU	HNIST	HTER	HMETEOR	Untrans. Words
DS	27.85 %	73.12 %	8.3602	20.74 %	82.80 %	2
PB	32.69 %	70.80 %	8.1710	23.63 %	86.00 %	34

Table 6.12: Precision grammar in-coverage comparison with state-of-the-art. Note: H-BLEU = BLEU against 150 post-edited MT output reference translations.

art.[10] For GCNs, however, the deep syntax system performs at least as well as the PB system by translating 56 out of 100 GCNs correctly and 25% in a way that adds some correct meaning to the translation (fuzzy), while the PB system translates 52% correctly and 22% as a fuzzy translation, in our human evaluation.

Table 6.12 contains results for the 38% of translations that were within coverage of the precision grammar used for generation, showing the PB system (32.69% BLEU) outperforming the deep syntax system (27.85% BLEU), by almost 5 BLEU points absolute. Due to the possibility of (ngram-based) BLEU unfairly biasing in favour of the PB system, we include results for human-targeted BLEU, NIST (Doddington, 2002), METEOR (Banerjee and Lavie, 2005) and TER (Snover et al., 2006, 2005) automatic evaluation metrics using reference translations produced by post-editing the first 150 translations from each MT system (Snover et al., 2006). Results for this evaluation show that the DS system (73.12% BLEU) in fact outperforms the PB system (70.8%) by a little over 2 BLEU points absolute for translations within coverage of the precision grammar used for generation. We also include the number of untranslated words for the deep syntax system (2 words) and the PB system (34 words), showing that for translations in-coverage of the precision grammar, the deep syntax system also achieves higher coverage of unseen data.

[10] The unfair bias of ngram-based BLEU metric in favour of Moses should be noted, and is discussed later.

Discussion

Automatic evaluation results for the entire test set suggest that our system underperforms significantly in comparison with state-of-the-art (Table 6.11). However, the results are unfairly biased in favour of the PB system, due to a combination of the BLEU evaluation metric being ngram-based with legitimate syntactic variations in the DS system output. The difference in results is, however, too large to claim that this is entirely due to this bias. Table 6.14 shows a random selection of translations produced by the DS system from the entire test set.

Human evaluation of 100 GCNs shows that the DS system does in fact achieve state-of-the-art performance for this particular syntactic construction, however. Interestingly, the intersection of the GCNs that the DS system translates correctly and the PB system is quite small, with our system correctly translating 30% of those not translated correctly by Moses, and Moses correctly translating 23% of those not translated correctly by our system, suggesting the possibility of a hybrid MT system (similar to (Eisele et al., 2008; Chen et al., 2007; Eisele, 2005)) or that deep syntax parsing could be used to improve translation of GCNs for PB-SMT. Table 6.13 shows a selection of GCNs taken from the entire test set for the PB and DS systems. The DS system achieves coverage of GCNs not observed in training data where component nouns were observed in training. For example, the GCN, *Hafenpolitik*, was not observed in the German training data, but *Hafen* appears combined with other nouns a total of approximately 80 times and *politik* also appears in the German training data approximately 3,400 times combined with another noun. This GCN is translated correctly by the deep syntax system but not the PB system.

For translations within coverage of the precision grammar, i.e. where the transfer decoder manages to produce a combination of lemmas, dependency relations and morpho-syntactic information in TL structures that do not clash with constraints during TL generation, human-targeted evaluation results show the DS system achieves state-of-the-art performance for these translations, in addition to achieving higher translation coverage of unseen data, mainly due to its ability to learn how

GCN	PB Translation	DS Translation
Wiederaufnahme	Resumption	
Tagesordnung	agenda	
Rechnungsführung	accounts	
Unternehmensneugründungen		company start-ups
Vorsichtsmassnahmen		measures precautionary*
Asien-Europa-Stiftung		Asia Europe Foundation
Osttimors		East Timor
ASEM-Gesprächen		ASEM talks
Hafenpolitik		port policy
Schwerpunkt	Emphasis	
Hauptsache	reason*	
Eigenkapital	capital*	invested capital*
Arbeitsrecht	labour law*	employment legislation*
Küstenstaaten		coastal states
Subsidiaritätsprinzip	principle of subsidiarity*	
Bewerberländer	candidate countries	applicant countries*
Parlamentswahlen	parliamentary elections*	general elections
Standpunkt	position*	question*
Ostsee	Baltic	
Änderungsantrag	Amendment	
Dioxinskandal	dioxin scare*	dioxin scandal
Einteilung	classification*	division
Futtermittelsicherheit		feed safety
Futtermittelkette		feed chain
Futtermitteln	feed*	means of feed*
Gemeinschaftsebene	Community level	Community scale*
Weltanschauung	World view*	world like mindedness*
weltweit	in the world*	worldwide*
Gemeinderatswahlen		elections local*
Richtlinien	directives*	directive*
Kernstück	heart*	lifeblood*
Ausnahmemöglichkeiten		opportunity for exceptions*
Änderungsanträgen	amendments	
Änderungsanträge		amendments
Vertragseinhaltung		Treaty compliance*
Entschliessungsantrags	resolution*	
Forschungsraum	research area	period of Research*
Endkontrolle		final*
Gegenprüfung		counter examination

Table 6.13: German Compound Noun translations for the Phrase-Based SMT system and the deep syntax system, translations evaluated as a fuzzy translation are marked with an asterisk

SRC:	Dies kann nicht hingenommen werden.
REF:	This is an unacceptable situation.
DS:	Not one that can allow continue
SRC:	Herr Präsident! Die Sicherheit verschiedener Verkehrsarten steht ernsthaft auf dem Spiel.
REF:	Mr President, safety is a serious issue for various forms of transport.
DS:	Mr President. Die of different forms of transport safety is at stake seriously.
SRC:	Das ist die politische Position.
REF:	That is the political position.
DS:	That is the political position.
SRC:	Natürlich ist sich auch die türkische Gesellschaft dieses Gegensatzes bewusst.
REF:	Turkish society obviously perceives this contradictory attitude.
DS:	Of course ist sich the Turkish society also of this contradiction bewusst
SRC:	Solche Gewalttätigkeit potenziert die Hassgefühle nur noch weiter.
REF:	That sort of violence only stirs up feelings of hatred.
DS:	This violation potenzieren only hate emotions further

Table 6.14: Randomly selected translations, original reference translations provided (not human-targeted)

to translate new unseen GCNs from GCNs in the training data that contain component nouns, in addition to achieving coverage of inflections of words not seen in bilingual training, since we use Factored Models (Koehn and Hoang, 2007). Table 6.15 shows a random selection of translations for the PB and DS systems for translations in coverage of the precision generation grammar and Table 6.16 shows German words that were not translated by the DS and PB systems for translations in coverage of the precision grammar.

An examination of the kinds of sentences that each system translates better or worse than the other showed that, in general, the DS system translates the following better than the PB system: compound nouns, the passive voice, the DS system does not tend to leave out nouns or verbs which are sometimes omitted by the PB system, does not omit determiners from nouns as often than the PB system, it produces more fluent verb forms e.g. *"contributes to achieving this objective"* was produced by the DS system as opposed to *"contributes to achieve this objective"* by the PB system.

SRC:	Auf Gesetzesebene gibt es allgemeine Texte, in denen Diskriminierung weltweit verurteilt wird.
REF:	**Legally speaking, there are general texts condemning discrimination everywhere.**
DS:	General texts condemning worldwide discrimination have been given to any legislative level.
PB:	There is general provisions on gesetzesebene where discrimination is condemned in the world.
SRC:	Das soll sich hier hoffentlich nicht wiederholen!
REF:	**I hope we will not see a repeat performance here!**
DS:	hopefully that should not be repeated.
PB:	I hope it will not repeat here!
SRC:	Der BSE-Skandal war das schlechteste, bekannteste Beispiel.
REF:	**The BSE scandal was the worst and most notorious example.**
DS:	The BSE scandal is the worst and most known case.
PB:	The BSE scandal was the worst and most well-known example.
SRC:	In Erwartung von mehr Klarheit haben wir uns deshalb der Stimme enthalten.
REF:	**Pending further clarification, we therefore abstain from the vote.**
DS:	Therefore I abstained in expectation of greater clarity for.
PB:	In expectation of greater clarity, we have therefore abstain from voting.
SRC:	Der Wiederaufbau Osttimors ist noch im Gange.
REF:	**The rebuilding of East Timor is still an ongoing process.**
DS:	The reconstruction of East Timor is still taking place.
PB:	The reconstruction osttimors is still in progress.
SRC:	Möchte sich jemand für diesen Antrag aussprechen?
REF:	**Is there a speaker to support this request?**
DS:	Does anyone wish to speak in support of this motion?
PB:	Does anyone wish to speak in favour of this request?
SRC:	Vielen Dank für diese Klarstellung, Herr Kommissar.
REF:	**Thank you very much for that clarification, Commissioner.**
DS:	I would like to thank the Commissioner for that clarification.
PB:	Thank you for that clarification, Commissioner.
SRC:	In diesem Punkt sind wir einer Meinung.
REF:	**On this point we are in agreement.**
DS:	We will be agreement on point about this.
PB:	In this regard, we are in agreement.
SRC:	Gibt es Einwände?
REF:	**Are there any comments?**
DS:	Are there any objections?
PB:	Are there any comments?
SRC:	Verhaltenskodex für Waffenausfuhren
REF:	**Arms trade code of conduct**
DS:	Code of Conduct on Arms Exports
PB:	Code of conduct on arms exports

Table 6.15: Randomly selected sample of translations in-coverage of precision grammar, original reference translations provided.

Phrase-Based System	Deep Syntax System
interparlamentarischer	liegen
asien-europa-stiftung	vorsichtshalber
osttimors	
interparlamentarischer	
europäers	
zu	
lehnten	
spielzeugbomben	
erfahrenen	
kompetenzverteilung	
marktposition	
enttäuschte	
selbstbewertung	
gegenwert	
kostengünstiges	
bleibenden	
geldverkehrs	
reformpläne	
eindämmungsmassnahmen	
regem	
auslandsdiplomatie	
kompetenzabgrenzung	
planungssicherheit	
papua-führer	
dominiert	
ersuchten	
neuzuteilung	
eu-lärmindizes	
zusatzstoffes	
klimafrage	
vorsichtshalber	
sicherheitsspielraum	
un-flüchtlingshilfswerk	
gesamtgesellschaftlichen	

Table 6.16: German words not translated in translations within coverage of the TL generation precision grammar for the Phrase-Based and deep syntax systems

Conversely, the DS system translates the following kinds of sentences worse than the PB system: the DS system can sometimes choose a influent verb-preposition combination or influent combinations of adverbs e.g. "addressed absolutely away" was produced by the DS system where the correct translation "raised" was produced by the PB system, the DS system can make a noun possessive when it should not be e.g. *“the focus's is ...”*, the DS system more often produces the incorrect tense for a verb, and also can produce influent adjectives for nouns, e.g. *“"expectations .. are large”* as opposed to *“expectations are high”*.

6.3 Deep Syntax Language Model Experiment

Deep syntax language models may not only be relevant to deep syntax transfer, but also have the potential to be integrated into other kinds of SMT systems. String-based and deep syntax language models both estimate the probability of a sentence by combining probabilities of individual words. For both types of model, the probability of a word is estimated using the probability of it occurring in a particular context. A traditional string-based language model uses the local context of each word within the string, specifically its preceding $n-1$ words, whereas a deep syntax language model ignores local context and instead uses as context the words that are linked to it via dependency relations, as described in Section 4.5.1. Ideally, both types of language model can be used in a single application to help produce output that is both fluent with respect to local combinations of words in the string and fluent with respect to combinations of words within the deeper structure. In the next section we highlight the potential of deep syntax language models for SMT systems in general followed by an experimental comparison of string-based and deep syntax language models. This work is also described in detail in Graham and van Genabith (2010).

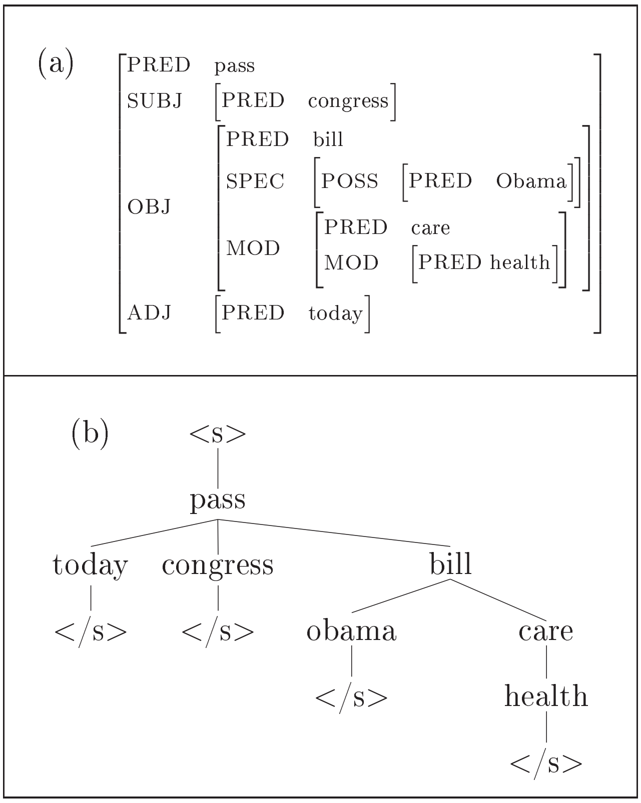


Figure 6.1: *"Today congress passed Obama's health care bill."*

6.3.1 Deep Syntax and Lexical Choice in SMT

Correct lexical choice in machine translation is extremely important and PB-SMT systems rely on the language model to ensure that when two phrases are combined with each other, the model can rank more fluent combinations of phrases higher than those that are less fluent. Conditioning the probability of each word on its deep context has the potential to provide a more meaningful context than the local context within the string. Figure 6.1 shows the LFG f-structure for English sentence *"Today congress passed Obama's health care bill."* [11] Encoded within the f-structure is a directed graph and our language model uses a simplified acyclic unlabeled version of this graph shown in Figure 6.1(b) within the f-structure of Figure 6.1(a). A comparison of the probabilities of individual words in the deep syntax model and string-based language model in Figure 6.2 highlights how the DS model may provide information to improve lexical choice for SMT systems. For instance,

[11] Morpho-syntactic information/ atomic features are omitted from the diagram.

(a) Deep Syntax LM	(b) Traditional LM
$p(e) \approx p(\text{ pass } \mid \text{<s>})*$	$p(e) \approx p(\text{ passed } \mid \text{ today congress })*$
$p(\text{ today } \mid \text{<s> pass })*$	$p(\text{ today } \mid \text{<s>})*$
$p(\text{</s>} \mid \text{ pass today })*$	
$p(\text{ congress } \mid \text{<s> pass })*$	$p(\text{ congress } \mid \text{<s> today })*$
$p(\text{</s>} \mid \text{ pass congress })*$	
$p(\text{ bill } \mid \text{<s> pass })*$	$p(\text{ bill } \mid \text{ health care })*$
$p(\text{ obama } \mid \text{ pass bill })*$	$p(\text{ obama } \mid \text{ congress passed })*$
$p(\text{</s>} \mid \text{ bill obama })*$	
$p(\text{ care } \mid \text{ pass bill })*$	$p(\text{ care } \mid \text{ s health })*$
$p(\text{ health } \mid \text{ bill care })*$	$p(\text{ health } \mid \text{ ' s })*$
$p(\text{</s>} \mid \text{ care health })$	
	$p(\text{ ' } \mid \text{ passed Obama })*$
	$p(\text{ s } \mid \text{ obama ' })*$
	$p(\text{ . } \mid \text{ care bill })*$
	$p(\text{</s>} \mid \text{ bill . })$

Figure 6.2: Example Comparison of Deep Syntax and Traditional Language Models

let us consider how the language model in a German to English SMT system is used to help rank the following two translations *today congress passed ...* and *today convention passed ...* (the word *Kongress* in German can be translated into either *congress* or *convention* in English). In the deep syntax model, the important competing probabilities are (i) $p(\textit{congress}|\text{<s>}\textit{pass})$ and (ii) $p(\textit{convention}|\text{<s>}\textit{pass})$, where (i) can be interpreted as the probability of the word *congress* modifying *pass* when *pass* is the head of the entire sentence and, similarly (ii) the probability of the word *convention* modifying *pass* when *pass* is the head of the entire sentence. In the traditional string-based language model, the equivalent competing probabilities are (i) $p(\textit{congress}|\text{<s>}\textit{today})$, the probability of *congress* following *today* when *today* is the start of the sentence and (ii) $p(\textit{convention}|\text{<s>}\textit{today})$, probability of convention following *today* when *today* is the start of the sentence, showing that the deep syntax language model is able to use more meaningful context for good lexical choice when estimating the probability of words *congress* and *convention* compared to the string-based language model.

In addition, the deep syntax language model will encounter less data sparseness problems for some words than a string-based language model. In many languages words occur that can legitimately be moved to different positions within the string without any change to dependencies between words. For example, sentential adverbs in English, can legitimately change position in a sentence, without affecting the underlying dependencies between words. The word *today* in *"Today congress passed Obama's health care bill"* can appear as *"Congress passed Obama's health care bill today"* and *"Congress today passed Obama's health care bill"*. Any sentence in the training corpus in which the word *pass* is modified by *today* will result in a bigram being counted for the two words, in a bigram deep syntax language model for example, regardless of the position of *today* within each sentence.

In addition, some surface form words such as auxiliary verbs are not represented as predicates in the deep syntactic structure. For lexical choice, it's not really the choice of auxiliary verbs that is most important, but rather the choice of an appropriate lexical item for the main verb (that belongs to the auxiliary verb). Including a model that ignores auxiliary verbs could aid better lexical choice, by focusing on the choice of a main verb without the effects of its auxiliary verb.

For some words, however, the probability in the string-based language model provides as good if not better context than the deep syntax model, but only for the few words that happen to be preceded by words that are important to its lexical choice, showing that the deep syntax language model should not replace the string-based model. For example, the probability of *bill* in Figures 6.2(a) and 6.2(b) is computed in the deep syntax model as $p(bill|$ <s> $pass)$ and in the standard model using $p(bill|health\ care)$, and for this word the local context seems to provide more important information than the deeper context when it comes to lexical choice. The deep model nevertheless adds some useful information, as it includes the probability of *bill* being an argument of *pass* when *pass* is the head of a sentence.

In string-based language modeling, the special start symbol is added at the beginning of a sentence so that the probability of the first word appearing as the first word of a sentence can be included when estimating the probability. With similar motivation, we add a start symbol to the deep syntactic representation so that the probability of the head of the sentence occurring as the head of a sentence can be included. For example, $p(be| \text{<s>})$ will have a high probability as the verb *be* is the head of many sentences of English, whereas $p(colorless| \text{<s>})$ will have a low probability since it is unlikely to occur as the head. We also add end symbols at the leaf nodes in the structure to include the probability of these words appearing at that position in a structure. For instance, a noun followed by its determiner such as $p(\text{</s>} |attorney\ a)$ would have a high probability compared to a conjunction followed by a verb $p(\text{</s>} |and\ be)$.

6.3.2 Evaluation

We carry out an experimental evaluation to investigate the potential of the deep syntax language model we describe in this thesis independently of any machine translation system. We train a 5-gram deep syntax language model on 7M English f-structures, and evaluate it by computing the perplexity and ngram coverage statistics on a held-out test set of parsed fluent English sentences. In order to provide an interesting comparison, we also train a traditional string-based 5-gram language model on the same data and test it on the the same held-out test set of English sentences. A deep syntax language model comes with the obvious disadvantage that any data it is trained on must be in-coverage of the parser, whereas a string-based language model can be trained on any available data of the appropriate language. Since parser coverage is not the focus of our work, we eliminate its effects from the evaluation by selecting the training and test data on the basis that they are in fact in-coverage of the parser.

6.3.3 Language Model Training

Our training data consists of English sentences from the WMT09 monolingual training corpus with sentence length range of 5-20 words that are in coverage of the

Corpus	Tokens	Ave. Tokens per Sent.	Vocab
Strings	138.6M	19	345K
LFG lemmas/predicates	118.4M	16	280K

Table 6.17: Language model tokens for deep syntax and string-based language models for the same training data of 7.29M sentences of Newswire text

parsing resources (Kaplan et al., 2004; Riezler et al., 2002) resulting in approximately 7M sentences. Preparation of training and test data for the string-based language model consisted of tokenization and lower casing. Parsing was carried out with XLE (Kaplan et al., 2002) and an English LFG grammar (Kaplan et al., 2004; Riezler et al., 2002). The parser produces a packed representation of all possible parses according to the LFG grammar and we select only the single best parse for language model training by means of a disambiguation model (Kaplan et al., 2004; Riezler et al., 2002). Ngrams were automatically extracted from the f-structures and lowercased. SRILM (Stolcke, 2002) was used to compute both language models. Table 6.17 shows statistics on the number of words and lemmas used to train each model.

6.3.4 Testing

The test set consisted of 789 sentences selected from WMT09 additional development sets[12] containing English Europarl text and again was selected on the basis of sentences being in-coverage of the parsing resources. SRILM (Stolcke, 2002) was used to compute test set perplexity and ngram coverage statistics for each order model.

Since the deep syntax language model adds end of sentence markers to leaf nodes in the structures, the number of (so-called) end of sentence markers in the test set for the deep syntax model is much higher than in the string-based model. We therefore also compute statistics for each model when end of sentence markers are omitted

[12] test2006.en and test2007.en

from both model training and testing.[13] In addition, since the vast majority of punctuation is not represented as predicates in LFG f-structures, we also test the string-based language model when punctuation has been removed.

6.3.5 Results

Table 6.18 shows perplexity scores and ngram coverage statistics for each order and type of language model. Note that perplexity scores for the string-based and deep syntax language models are not directly comparable, because although trained on the same set of sentences, the data is in a different format for each model, lemmas for the deep syntax model and surface form words for the string-based model, so each model in fact has a different vocabulary. Ngram coverage statistics provide a better comparison.

Unigram coverage for all models is high as each achieves close to 100% coverage on the held-out test set. Bigram coverage is highest for the deep syntax language model when end of sentence *eos* markers are included (94.71%) with next highest coverage achieved by the string-based model that also includes *eos* markers (93.09%). When *eos* marker probabilities are omitted bigram coverage goes down slightly to 92.44% for the deep syntax model and to 92.83% for the string-based model, and when punctuation is also omitted from the string-based model, coverage goes down again to 91.57%.

Trigram coverage statistics for the test set maintain the same rank between models as in the bigram coverage, from highest to lowest as follows: DS+eos at 64.71%, SB+eos at 58.75%, SB-eos at 56.89%, DS-eos at 53.67%, SB-eos-punc at 53.45%. For 4-gram and 5-gram coverage a similar coverage ranking is seen, but with DS-eos (4gram at 17.17%, 5gram at 3.59%) and SB-eos-punc (4gram at 20.24%, 5gram at 5.76%) swapping rank position.

[13] When we include end of sentence marker probabilities we also include them for normalization, and omit them from normalization when their probabilities are omitted.

	1-gram		2-gram		3-gram		4-gram		5-gram	
	cov.	ppl	cov.	ppl	cov.	ppl	cov.	ppl	cov.	ppl
SB-eos	99.61%	1045	92.83%	297	56.89%	251	23.32%	268	7.19%	279
SB-eos-punc	99.58%	1357	91.57%	382	53.45%	327	20.24%	348	5.76%	360
DS-eos	99.56%	1005	92.44%	422	53.67%	412	17.17%	446	3.59%	453
SB + eos	99.63%	900	93.09%	227	58.75%	194	25.48%	207	8.35%	215
DS + eos	99.70%	211	94.71%	77	64.71%	73	29.86%	78	8.75%	79

Table 6.18: Ngram coverage and perplexity (ppl) on held-out test set. Note: DS = deep syntax, SB string-based, eos = end of sentence markers

6.3.6 Discussion

Ngram coverage statistics for the DS-eos and SB-eos-punc models provide the fairest comparison, and the deep syntax model achieves similar coverage to the string-based model, with the deep syntax model achieving higher coverage than the string-based model for bigrams (+0.87%) and trigrams (+0.22%), marginally lower coverage coverage of unigrams (-0.02%) and lower coverage of 4-grams (-3.07%) and 5-grams (2.17%) compared to the string-based model.

Perplexity scores for the deep syntax model when probabilities of *eos* symbols are included are low (79 for the 5gram model) and this is caused by *eos* markers in the test set in general being assigned relatively high probabilities by the model, and since several occur per sentence, the perplexity increases considerably when their probabilities are omitted (453 for the 5gram model).

Tables 6.19 and 6.20 show the most frequently encountered trigrams in the test set for each type of model. A comparison shows how different the two models are and highlights the potential of the deep syntax language model to aid lexical choice in SMT systems. Many of the most frequently occurring trigram probabilities for the deep syntax model are for arguments of the main verb of the sentence, conditioned

3-gram	No. Occ.	Prob.
<s> and be	42	0.1251
<s> be this	21	0.0110
<s> must we	19	0.0347
<s> would i	19	0.0414
<s> be in	17	0.0326
<s> be that	14	0.0122
be debate the	13	0.0947
<s> be debate	13	0.0003
<s> can not	12	0.0348
<s> and president	11	0.0002
<s> would like	11	0.0136
<s> would be	11	0.0835
<s> be also	10	0.0075

Table 6.19: Most frequent trigrams in test set for deep syntax model

on the main verb, and including such probabilities in a translation model could improve fluency as information about which words are in a dependency relation together is explicitely included in the model. In addition, a frequent trigram in the held-out data is *<s> be also*, where the word *also* is a sentential adverb modifying *be*. Trigrams for sentential adverbs are likely to be less effected by data sparseness in the deep syntax model compared to the string-based model which could result in the deep syntax model improving fluency with respect to combinations of main verbs and their modifying adverbs. The most frequent trigram in the deep syntax test set is *<s> and be*, in which the head of the sentence is the conjunction *and* with argument *be*. In this type of syntactic construction in English, it's often the case that the conjunction and verb will be distant from each other in the sentence, for example: *Nobody was there except the old lady and without thinking we quickly left.* (where *was* and *and* are in a dependency relation). Using a deep syntax language model could therefore improve lexical choice for such words, since they are too distant for a string-based model.

3-gram	No. Occ.	Prob.
mr president ,	40	0.5385
<s> this is	25	0.1877
by the european	20	0.0014
the european union	18	0.1096
<s> it is	16	0.1815
the european parliament	15	0.0252
would like to	15	0.4944
<s> i would	15	0.0250
<s> that is	14	0.1094
i would like	14	0.0335
and gentlemen ,	13	0.1005
ladies and gentlemen	13	0.2834
<s> we must	12	0.0120
should like to	12	0.1304
i should like	11	0.0089
, ladies and	11	0.5944
, it is	10	0.1090

Table 6.20: Most frequent trigrams in test set for string-based model

6.4 Conclusion

A detailed evaluation of an SMT via deep syntactic transfer system was presented. Experimental results show that the deep syntax intersection word alignment method achieves by far the best results for the system, with larger rule size limits also improving results. Varying the beam size does not have a dramatic effect on MT performance, with a beam size as low as 10 being sufficient for the system. In addition, significant gains can be made by increasing the size of the m-best decoder output list to 100 and with non-deterministic generation. Compared to state-of-the-art PB-SMT the deep syntax system under-performs, but for sentences in-coverage of the precision grammar used for generation, state-of-the-art performance and higher coverage of unseen data is achieved.

We also presented a comparison of a deep syntax and traditional string-based language model. Results showed that the deep syntax language model achieves similar ngram coverage to the string-based model on a held out test set. We highlighted the potential of integrating such a model into SMT systems for improving lexical choice by using a deeper context for probabilities of words compared to a string-based model.

Chapter 7

Conclusions and Future Work

This thesis presented an investigation into an approach to machine translation that integrates state-of-the-art PB-SMT techniques into a deep syntactic transfer architecture. We described methods of automatically word/node aligning deep syntactic structures, as well as transfer rule extraction. We developed a new definition for consistent transfer rules, inspired by the definition of a consistent phrase in PB-SMT (Koehn et al., 2003). Similar to phrase extraction in PB-SMT we extract all transfer rules consistent with the word/node alignment. Since we allow non-terminals in transfer rules, this can result in large numbers of rules, and we provide a new method of efficiently extracting and storing transfer rules, as well as releasing the rule extraction software as an open source tool. Our experimental evaluation showed that including larger transfer rules that contain more context as well as smaller rules is indeed worthwhile as it results in substantially better MT output.

The thesis also presented the design and implementation of a deep syntax transfer decoder, and we provide this tool to the wider research community as open source software to assist future research. Our translation model, a log-linear combination of feature functions, includes a trigram deep syntax language model, which is fully and efficiently integrated into decoding. We also described a new method of translating morpho-syntactic information, factor templates, used to decide which morpho-syntactic factors to translate separately from lemmas in Factored Models, which significantly improves MT output for our system.

We finally presented a detailed evaluation of the machine translation system, in which we investigate the effects of using different methods of word alignment, different beam sizes during transfer decoding, generating from different sized m-best decoder output lists, using deterministic versus non-deterministic generation, as well as comparing the current performance of the system with a state-of-the-art PB-SMT system. Results showed that although the deep syntax system does not achieve state-of-the-art performance for the entire test set, for sentences within coverage of the precision grammar used for generation, state-of-the-art performance is achieved. In addition, a manual evaluation of the translation of German compound

	Riezler & Maxwell (2006)	Bojar & Hajic (2008)	Graham (2010)
Lemmatized Word Alignment	✗	✓	✓
DS Reordered Word Alignment	✗	✗	✓
Train Decoder Fully Automatically	✗	✓	✓
Non-isomorphism	✗	✗	✓
LM during decoding	✗	✓	✓
Factored Models	✗	✓	✓
Factor Templates	✗	✗	✓
Unlimited Rule Size	✗	✗	✓

Table 7.1: Summary of Contrasts with Related Work

nouns, revealed the deep syntax system achieves state-of-the-art performance for automatic translation of this particular syntactic construction on the entire test set. Finally, we provided a comparison of the deep syntax language model we use in our work with a traditional string-based language model, in order to highlight its potential to improve lexical choice in general in SMT systems. Table 7.1 provides a summary of the contributions in respect to how they compare to related research.

7.1 Research Questions & Motivations

This thesis investigated the research questions detailed in Section 1.4. The main challenges of using deep syntax for transfer in machine translation was a main research question. In our investigation, the most significant challenge identified for the MT approach was the challenge of automatically constructing TL deep syntactic structures that do not cause generation clashes and this remains a significant challenge. The reason for this is due to the large number of possible combinations of TL lexical items, dependency relations, and values of atomic features. We believe an integration of the target language grammar rules used for generation into transfer decoding could greatly increase the proportion of grammatical structures produced by the decoder. Due to time constraints, we leave this for future work. An additional aim was to apply machine learning methods to deep syntactic transfer. The work stayed true to this aim as all of the methods of training that were developed were fully automatic. The methods described in this thesis achieve a high level of language pair independence, since none of the methods are specific to any particular language pair, in addition to the methods being linguistic theory independent (within the context of deep syntax), as all of the methods described here can be applied to other theories of deep syntax with little to no adaptation required. We achieved the aim of applying Phrase-Based SMT methods to deep syntactic transfer since our rule extraction method is very similar to the way in which phrases are extracted in PB-SMT, in addition to our translation model being a log-linear combination of feature functions that includes several features adopted from PB-SMT.

We also wished to develop efficient methods of training and decoding, which was achieved, as all of the methods developed are efficient and can scale to large corpora. As part of the work we have also made the two main tools open source to aid future research, the transfer decoder and the rule extraction software. In addition, we investigated the effects of system parameters on translation quality providing insight into which parameters are of greatest importance to translation quality. Finally, we provided an empirical comparison of deep syntactic transfer and Phrase-Based SMT.

7.2 Future Work

SMT via deep syntactic transfer is an approach that does not have many theoretical flaws, as a system with this architecture can in theory translate cross-lingual language phenomena that cause significant challenges for other MT approaches, such as long-distance dependencies between words, as well as achieving a high level of language pair independence as no reordering model is required. The main challenges the approach is faced with are of a more practical nature compared to other approaches to MT.

For parsing and generation, higher parser coverage of training data is needed so that all available bilingual training data can be used. Similarly, increased generator coverage and robustness is also needed. We have shown that when TL structures output by the decoder fall within coverage of the precision grammar for generation, that the quality of the MT output is high. Achieving a good combination of TL lemmas, dependency relations and morpho-syntactic information, we believe, is the most significant challenge for this approach. The very large number of possible combinations of lemmas, dependency relations and morpho-syntactic information in a single TL structure makes obtaining a combination that does not cause clashes in generation extremely difficult. In our evaluation, our system managed to achieve no generation clashes for approximately 38% of the test set. A possible way to increase this would be to use information from the precision grammar and lexicon during transfer decoding.

Increased parser and generator coverage could be achieved by employing fully statistical resources, like those described in Section 2.2.2. Such statistical technologies do not produce as fine-grained an analysis as the precision grammar, however, and its difficult to know what kind of effect this will have on MT output. On the one hand, its possible that the coarser analysis omits atomic features that are important for translation resulting in a decrease in performance. On the other hand, however, if atomic features are present in the hand-crafted grammar analysis that are not in fact needed for translation, which is quite possible, removing them from the analysis removes the need for the system to accurately guess their values in the TL structure and could result in less generator clashes and fewer good translations being pruned prior to generation.

Although a statistical generator is likely to increase robustness of generation, there is also the possibility, however, that the high level of grammaticality achieved when sentences are within coverage of the precision grammar will be lost. In addition, since these technologies are usually tested on gold-standard input, its likely that significant modification will be required before they can be used for lower quality input, such as the transfer decoder output structures.

Other possibilities for future work include the development of better word alignment methods. Our method of word alignment does not explicitely use some of the information present in deep syntactic structures, like the position of nodes in the underlying graph structure. For example, its likely that within the SL structure nodes positioned close to the root are aligned with similar positioned nodes in the TL structure, and our method does not explicitely use this kind of information. In addition, the grammatical function of a word may in some cases provide more useful information than the lexical item and could be taken advantage of for automatic word alignment. For example, a better way of aligning a determiner or adjunct might be to omit it from the training data in a first-stage alignment, then use the alignment of its head to find the word its aligned to. In addition, the transfer decoder could be improved as it currently does not include hypothesis recombination or future cost estimation, which could potentially improve the search.

Some of the methods developed in this work could be adapted to a PB-SMT architecture and potentially improve such systems. The deep syntax language model has the potential to improve lexical choice in a PB-SMT system if successfully integrated. In addition, it would be interesting to investigate if factor templates can be used to improve PB-SMT. We showed how they significantly improve results within a deep syntax transfer architecture, and although the architectures are very different especially considering that the deep syntax architecture restricts us to using sentence-level generation, as opposed to the word-level generation of Factored Phrase-Based Models, they still could potentially provide a kind of halfway house between standard PB-SMT and fully Factored Models in addition to providing a way of accurately translating idiosyncratic translations. In addition, the deep analysis of compound nouns provided by the deep syntax parser could be taken advantage of in PB-SMT, for German at least. German training and test data could be parsed thus splitting German compound nouns into component nouns in a preprocessing step. This could add to a PB-SMT system the ability to learn unseen German compound nouns from component nouns observed in the training data.

Bibliography

Aranberri-Monasterio, N. and O'Brien, S. (2009). Evaluating rbmt output for -ing forms: A study of four target languages. *Linguistica Antverpiensia*, 8:105–122.

Avramidis, E. and Koehn, P. (2008). Enriching morphologically poor languages for statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies*, pages 763–770, Columbus, Ohio.

Banerjee, S. and Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgements. In *Proceedings of Workshop on "Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, at the 43rd Annual Meeting of the Association for Computational Linguistics"*, pages 65–73, Ann Arbor, Michigan.

Birch, A., Osborne, M., and Koehn, P. (2007). CCG supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation at the 45th Annual Meeting of the Association for Computational Linguistics*, pages 9–16, Prague, Czech Republic. Association of Computational Linguistics.

Bojar, O. (2009). *Exploiting Linguistic Data in Machine Translation.* PhD thesis, Ufal, Charles University, Prague.

Bojar, O. and Hajič, J. (2008). Phrase-Based and Deep Syntactic English-to-Czech Statistical Machine Translation. In *Proceedings of the third Workshop on Sta-*

tistical Machine Translation at the 46th Annual Meeting of the Association for Computational Linguistics, Columbus, Ohio.

Bojar, O. and Čmejrek, M. (2007). Mathematical Model of Tree Transformations. In *Project Euromatrix Deliverable 3.2*, Ufal, Charles University, Prague.

Bresnan, J. (2001). *Lexical-Functional Syntax.* Blackweel, Oxford.

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

Brown, P., Cocke, J., Pietra, S., Pietra, V., Jelinek, F., Mercer, R., and Roosin, P. (1990). A statistical approach to machine translation. *Computational Linguistics*, pages 79–85.

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

Buch-Kromann, M. (2007). Computing translation units and and quantifying parallelism in dependency treeebanks. In *Proceedings of the Linguistics Annotation Workshop*, pages 69–76, Prague, Czech Republic.

Burke, M., Cahill, A., O'Donovan, R., van Genabith, J., and Way, A. (2004a). Treebank-based acquisition of wide-coverage, probabilistic lfg resources: Project overview, results and evaluation. In *Proceedings of The First International Joint Conference on Natural Language Processing (IJCNLP-04), Workshop Beyond shallow analyses - Formalisms and statistical modeling for deep analyses*, Sanya City, Hainan Island, China.

Burke, M., Lam, O., Cahill, A., Chan, R., O'Donovan, R., Bodomo, A., van Genabith, J., and Way, A. (2004b). Treebank-based acquisition of a chinese lexical-functional grammar. In *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation*, pages 161–172, Tokyo, Japan.

Cahill, A. (2004). *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations.* PhD thesis, Dublin City University.

Cahill, A., Burke, M., O'Donovan, R., Riezler, S., van Genabith, J., and Way, A. (2008). Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34:81–124.

Cahill, A., Burke, M., O'Donovan, R., van Genabith, J., and Way, A. (2004). Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-based LFG approximations. In *Proceedings of the 42nd Annual Meeting of the Association of Computational Linguistics.*

Cahill, A., Forst, M., Burke, M., McCarthy, M., O'Donovan, R., Rohrer, C., van Genabith, J., and Way, A. (2005). Treebank-based acquisition of multilingual unification grammar resources. *Journal of Research on Language and Computation; Special Issue on Shared Representations in Multilingual Grammar Engineering*, pages 247–279.

Cahill, A., McCarthy, M., van Genabith, J., and Way, A. (2002a). Automatic annotation of the penn-treebank with lfg f-structure information. In *Proceedings of the Language Resources and Evaluation Conference Workshop on Linguistic Knowledge Acquisition and Representation - Bootstrapping Annotated Language Data, Third International Conference on Language Resources and Evaluation*, pages 8–15, Paris, France. ELRA - European Language Resources Association.

Cahill, A., McCarthy, M., van Genabith, J., and Way, A. (2002b). Parsing with pcfgs and automatic f-structure annotation. In *Proceedings of the Seventh International Lexical Functional Grammar Conference*, pages 76–95, Stanford, California. CSLI Publications.

Cahill, A. and van Genabith, J. (2006). Robust PCFG-Based Generation using Automatically Acquired LFG Approximations. In *Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 1033–1040, Sydney, Australia.

Carl, M. (2007). METIS-II: The German to English MT System. In *Proceedings of the Machine Translation Summit XI*.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of the Association of Computational Linguistics*, Ann Arbor, Michigan.

Charniak, E., Knight, K., and Yamada, K. (2003). Syntax-based Language Models for Statistical Machine Translation. In *Proceedings of the Machine Translation Summit IX 2003*, New Orleans, Louisiana.

Chen, B., Cettolo, M., and Federico, M. (2006). Reordering rules for phrase-based statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, Kyoto, Japan.

Chen, Y., Eisele, A., Federmann, C., Hassler, E., Jellinghaus, M., and Theison, S. (2007). Multi-engine machine translation with an open-source SMT deocder. In *Proceedings of the Second Workshop on Statistical Machine Translation at the 45th Annual Meeting of the Association for Computational Linguistics*, pages 193–196, Prague, Czech Republic.

Chiang, D. (2007a). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, Ann Arbor, Michigan. Association for Computational Linguistics.

Chiang, D. (2007b). Hierarchical Phrase-based Models of Translation. *Computational Linguistics*, 2.

Chrupala, G. (2008). *Towards a Machine-Learning Architecture for Lexical Functional Grammar Parsing.* PhD thesis, Dublin City University.

Chrupala, G., Stroppa, N., van Genabith, J., and Dinu, G. (2007). Better training for function labeling. In *Proceedings of the Recent Advances in Natural Language Processing Conference*, Borovets, Bulgaria.

Chrupala, G. and van Genabith, J. (2006). Improving treebank-based automatic lfg induction for spanish. In *Proceedings of the 11th International Lexical Functional Grammar Conference*, Konstanz, Germany. CSLI Publications.

Chrupala, G. and van Genabith, J. (2007). Using very large corpora to detect raising and control verbs. In *Proceedings of the 12th International Lexical Functional Grammar Conference*, Stanford, California. CSLI Publications.

Corston-Oliver, S. and Gammon, M. (2004). Normalizing German and English inflectional morphology to improve statistical machine translation. In *Proceedings of the 6th Association of Machine Translation in the Americas Conference (AMTA 2004)*, pages 48–57, Washinton DC.

Costa-jussa, M. R. and Fonollosa, J. A. (2006). Statistical Machine Reordering. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 70–76, Sydney, Australia.

Crego, J. M. and Habash, N. (2008). Using shallow syntax information to improve word alignment and reordering for SMT. In *Proceedings of the Third Workshop on Statistical Machine Translation at the 46th Annual Meeting of the Association of Compuational Linguistics: Human Language Technologies*, pages 53–61, Columbus, Ohio.

Crego, J. M. and Marino, J. B. (2006). Integration of POS-tag-based source reordering into SMT decoding by an extended search graph. In *Proceedings of the 5th Conference of the Association of Machine Translation in the Americas (AMTA)*, Boston, Massachusetts.

Crego, J. M. and Marino, J. B. (2007). Syntax-enhanced n-gram based SMT. In *Proceedings of the Machine Translation Summit XI*, Copenhagen, Denmark.

Dalrymple, M. (2001). *Lexical-Functional Grammar.* Academic Press, San Diego, CA; London.

Deneefe, S. and Knight., K. (2009). Synchronous Tree Adjoining Machine Translation. In *Proceedings of Empirical Methods in Natural Language Processing Conference.*

Deneefe, S., Knight, K., Wang, W., and Marcu, D. (2007). What can Syntax-based MT learn from Phrase-based MT? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processsing and Computational Natural Language Learning.*

Doddington, G. (2002). Automatic Evaluation of Machine Translation Quality using N-gram Co-Occurrence Statistics. In *Proceedings of Human Languages Technologies Conference*, San Diego, California.

Dreyer, M., Hall, K., and Khudanpur, S. (2007). Comparing reordering constraints for SMT using efficient BLEU oracle computation. In *Proceedings of Syntax and Structure in Statistical Translation Workshop at Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics/ Association for Machine Translation in the Americas*, pages 103–110, Rochester, New York.

Eisele, A. (2005). First steps towards multi-engine machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Compuational Linguistics Workshop on Building and Using Parallel Texts*, pages 155–158, Ann Arbor, Michigan.

Eisele, A., Federmann, C., Saint-Amand, H., Jellinghaus, M., Herrmann, T., and Chen, Y. (2008). Using Moses to integrate multiple rule-based machine translation engines into a hybrid system. In *Proceedings of the Third Workshop on*

Statistical Machine Translation at the 46th Annual Meeting of the Association for Compuational Linguistics, pages 179–182, Columbus, Ohio.

Eisner, J. (2003). Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 205–208, Sappora.

Forst, M. (2007). *Disambiguation for a Linguistically Precise German Parser.* PhD thesis, University of Stuttgart.

Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, Boston, MA.

Gragg, G. B. (1995). *Babylonian Grammatical Texts*, pages 20–21. Elsevier Science Ltd.

Graham, Y. (2010). Sulis: An open source transfer decoder for deep syntactic statistical machine translation. *In The Prague Bulletin of Mathematical Linguistics, Special Issue: Open Source Tools for Machine Translation.*

Graham, Y., Hogan, D., and van Genabith, J. (2007). Automatic evaluation of generation and parsing for machine translation with automatically acquired transfer rules. In *Proceedings of the 2007 Workshop on Using Corpora for Natural Language Generation: Language Generation and Machine Translation at MT Summit XI*, Copenhagen.

Graham, Y. and van Genabith, J. (2008). Packed rules for automatic transfer rule induction. In *Proceedings of the European Association of Machine Translation Conference 2008*, Hamburg, Germany.

Graham, Y. and van Genabith, J. (2009). An open source rule induction tool for transfer-based smt. *The Prague Bulletin of Mathematical Linguistics, Special Issue: Open Source Tools for Machine Translation*, pages 37–46.

Graham, Y. and van Genabith, J. (2010). Deep syntax language models and statistical machine translation. In *Proceedings of the Fourth International Workshop on Syntax and Structure in Statistical Translation at The 23rd International Conference on Computational Linguistics*, Beijing, China.

Guo, Y. (2009). *Treebank-based Acquisition of Chinese Lexical Functional Grammar Resources for Parsing and Generation*. PhD thesis, Dublin City University.

Guo, Y., van Genabith, J., and Wang, H. (2007a). Acquisition of wide-coverage, robust, probabilistic lexical-functional grammar resources for chinese. In *Proceedings of the 12th International Lexical Functional Grammar Conference (LFG 2007)*, pages 214–232, California, USA. CSLI Publications.

Guo, Y., van Genabith, J., and Wang, H. (2008a). Dependency-based n-gram models for general purpose sentence realisation. In *Proceedings of the 22th International Conference on Computational Linguistics (COLING 2008)*, pages 297–304, Manchester, UK.

Guo, Y., Wang, H., and van Genabith, J. (2007b). Recovering non-local dependencies for chinese. In *Proceedings of the 2007 Joint Meeting of the Conference on Empirical Methods in Natural Language Processing (EMNLP) and the Conference on Computational Natural Language Learning (CoNLL)*, pages 257–266, Prague, Czech Republic.

Guo, Y., Wang, H., and van Genabith, J. (2008b). Accurate and robust lfg-based generation for chinese. In *Proceedings of the 5th International Natural Language Generation Conference (INLG 08)*, pages 86–94, Ohio, USA.

Hajič, J., Čmejrek, M., Dorr, B., Ding, Y., Eisner, J., Gildea, D., Koo, T., Parton, K., Penn, G., Radev, D., and Rambow, O. (2002). Natural Language Generation in the Context of Machine Translation. In *NLP WS'02, final report.*

Hogan, D., Cafferkey, C., Cahill, A., and van Genabith, J. (2007). Exploiting Multi-Word Units in History-Based Probabilistic Generation. In *Proceedings of the Empirical Methods in Natural Language Processing Conference.*

Judge, J., Burke, M., Cahill, A., O'Donovan, R., van Genabith, J., and Way, A. (2005). Strong domain variation and treebank-induced lfg resources. In *Proceedings of the Tenth International Conference on Lexical Functional Grammar*, pages 186–204, Bergen, Norway.

Kaplan, R. M. (1995). The formal architecture of lexical functional grammar. In Dalrymple, M., editor, *Formal Issues in Lexical Functional Grammar*, pages 7–28, Stanford, CA. CSLI Publications.

Kaplan, R. M. and Bresnan, J. (1982). Lexical Functional Grammar, a Formal System for Grammatical Representation. In Bresnan, J., editor, *The Mental Representation of Grammatical Relations*, pages 173–281.

Kaplan, R. M., King, T. H., and Maxwell, J. T. (2002). Adapting existing grammars: the XLE experience. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

Kaplan, R. M. and Maxwell, J. T. (1996). LFG Grammar Writer's Workbench. Technical report, Xerox Parc.

Kaplan, R. M., Riezler, S., King, T. H., Maxwell, J. T., and Vasserman, A. (2004). Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics Meeting*, Boston.

Klein, D. and Manning, C. D. (2003a). Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430.

Klein, D. and Manning, C. D. (2003b). Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pages 3–10, Cambridge, MA. MIT Press.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the tenth Machine Translation Summit*, Phuket, Thailand.

Koehn, P. (2009). *Statistical Machine Translation*, chapter 5. Cambridge University Press.

Koehn, P. and Hoang, H. (2007). Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 868–876, Prague.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Hoang, E. H. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session*, Prague, Czech Republic.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of Human Language Technology - North American Chapter of the Association for Computational Linguistics Conference*, pages 48–54, Edmonton, Alberta.

Langkilde, I. (2000). Forest-based statistical sentence generation. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 170–177, Seattle, Washington.

Li, Z., Callison-Burch, C., Khundanpur, S., and Thorton, W. (2009). Decoding in Joshua, Open Source Parsing-Based Machine Translation. In *The Prague Bulletin of Mathematical Linguistics, Special Issue: Open Source Tools for Machine Translation*, volume 91, pages 47–56.

Liu, Y., Qun, L., and Lin, S. (2006). Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Compuational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, Sydney, Australia.

Marcu, D., Wang, W., Abdessamad, E., and Knight, K. (2006). SPMT statistical machine translation with syntactic target language phrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sydney, Australia.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 110–115, Princeton, NJ.

Marcus, M. P., Santorini, B., and Marcinkiewivz, M. A. (1993). Building a large annotated corpus of english. *Computational Linguistics*, 19.

Maxwell, J. T. and Kaplan, R. (1993). The interface between phrasal and functional constriants. *Computational Linguistics*, pages 571–590.

Maxwell, J. T. and Kaplan, R. M. (1991). A Method for Disjunctive Constraint Satisfaction. In Tomita, M., editor, *Current Issues in Parsing Technology*, pages 173–190. Kluwer Academic Publishers.

Maxwell, J. T. and Kaplan, R. M. (1996). An efficient parser for lfg. In *Proceedings of the First International Lexical Functional Grammar Conference*, Grenoble, France. CSLI Publications.

Menezes, A. and Richardson, S. D. (2001). A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora. In *Proceedings of Workshop on Data-Driven Machine Translation at the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Och, F. J. and Ney., H. (2002). Discriminative training and maximum entropy models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, PA.

Och, F. J., Tillmann, C., and Ney, H. (1999). Improved alignment models for statistical machine translation. In *Proceedings of the 1999 Conference on Empirical Methods in Natural Language Processsing EMNLP 99*, pages 20–28, College Park, MD.

O'Donovan, R. (2006). *Automatic Extraction of Large-scale Multilingual Lexical Resources*. PhD thesis, Dublin City University.

O'Donovan, R., Burke, M., Cahill, A., van Genabith, J., and Way, A. (2004). Large-scale induction and evaluation of lexical resources from the penn-ii treebank. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 368–375, Barcelona, Spain.

O'Donovan, R., Burke, M., Cahill, A., van Genabith, J., and Way, A. (2005a). Large-scale induction and evaluation of lexical resources from the penn-ii and penn-iii treebanks. *Computational Linguistics*, pages 329–366.

O'Donovan, R., Cahill, A., van Genabith, J., and Way, A. (2005b). Automatic acquisition of spanish lfg resources from the cast3lb treebank. In *Proceedings of the Tenth International Conference on LFG*, pages 334–352, Bergen, Norway.

Owczarzak, K. (2008). *A Novel Dependency-Based Evaluation Metric for Machine Translation*. PhD thesis, Dublin City University.

Owczarzak, K., Graham, Y., and van Genabith, J. (2007a). Using f-structures in machine translation evaluation. In *Proceedings of the Lexical Functional Grammar*, Stanford, California. CSLI Publications.

Owczarzak, K., van Genabith, J., and Way, A. (2007b). Dependency-based Automatic Evaluation for Machine Translation. In *Proceedings of the 1st Workshop on Syntax and Structure in Statistical Translation at Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics.*

Owczarzak, K., van Genabith, J., and Way, A. (2007c). Labelled dependencies in machine translation evaluation. In *Proceedings of the Second Workshop on Statistical Machine Translation at the 45th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Prague, Czech Republic.

Owczarzak, K., van Genabith, J., and Way, A. (2008). Evaluating machine translation with lfg dependencies. *Machine Translation*, pages 95–119.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2001). BLEU: a Method for Automatic Evaluation of Machine Translation. Technical report, IBM T.J. Watson Research Center, Yorktown Heights, NY.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia.

Rehbein, I. (2009). *Treebank-based German Grammar Acquisition.* PhD thesis, Dublin City University.

Riezler, S., King, T. H., Kaplan, R. M., Crouch, R., Maxwell, J. T., and Johnson, M. (2002). Parsing the Wall Street Journal using Lexical Functional Grammar and discriminitive estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia.

Riezler, S. and Maxwell, J. (2006). Grammatical Machine Translation. In *Proceedings of Human Language Technologies and the 44th Annual Meeting of the Assciation for Computational Linguistics*, pages 248–255, New York.

Sgall, P., Hajicova, E., and Panevova, J. (1986). *The Meaning of the Sentence and its Semantic and Pragmatic Aspects.* Dordrecht: Reidel and Prague: Academia.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:50–64.

Snover, M., Dorr, B., Scwartz, R., Makhoul, J., and Micciula, L. (2006). A Study of Translation Error Rate with Targeted Human Annotation. In *Proceedings of the 7th biennial Conference of the Association for Machine Translaiton in the Americas*, pages 223–231, Boston, Massachusetts.

Snover, M., Dorr, B., Scwartz, R., Makhoul, J., Micciula, L., and Weischeidel, R. (2005). A Study of Translation Error Rate with Targeted Human Annotation. Technical report, University of Maryland, College Park, MD.

Stolcke, A. (2002). Srilm - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, Denver, Colorado.

Uszkoreit, H. (2009). Linguistics in computational linguistics: Observations and predictions. In *Proceedings of the European Chapter of Association for Computational Linguistics Workshop on the Interaction between Linguistics and Computational Linguistics*, pages 22–25, Athens, Greece. Association for Computational Linguistics.

Čmejrek, M. (2006). *Using dependency tree structure for Czech-Englsih machine translation*. PhD thesis, Ufal, MFF UK, Prague, Czech Republic.

Zaidan, O. (2009). Z-mert: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics, Special Issue: Open Source Tools for Machine Translation*, pages 79–88.

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