Learning Transfer Rules for Machine Translation with Limited Data

A Dissertation
submitted to the graduate school
in partial fulfillment of the requirements
for the degree
Doctor or Philosophy
in Language and Information Technologies
by
KATHARINA PROBST

Committee:
Alon Lavie (chair)
Jaime Carbonell
Lori Levin
Bonnie Dorr, University of Maryland, College Park

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania, U.S.A.

August 15, 2005

Abstract

The transfer-based approach to machine translation (MT) captures structural transfers between the source language and the target language, with the goal of producing grammatical translations. The major drawback of the approach is the development bottleneck, requiring many human-years of rule development. On the other hand, data-driven approaches such as example-based and statistical MT achieve fast system development by deriving mostly non-structural translation information from bilingual corpora. This thesis aims at striking a balance between both approaches by inferring transfer rules automatically from bilingual text, aiming specifically at scenarios where bilingual data is in sparse supply. The rules are learned using a variety of information, such as parses that are available for one of the languages, and morphological information that is available for both languages. They are learned in three stages, first producing an initial hypothesis, then capturing the syntactic structure, and finally adding appropriate unification constraints. The learned rules are used in a run-time translation system, a statistical transfer system which is a combination of a transfer engine and a statistical decoder. We demonstrate the effectiveness of the learned rules on Hebrew→English and a Hindi→English translation tasks.

The main contribution of this thesis is a new framework for inferring structural information with feature constraints from bilingual text, as well as an investigation of the taxonomy of learnable rules and their effectiveness. The framework is designed to be applicable for any language pair, and the inferred rules can be used in conjunction with a statistical decoder. In addition to presenting methods to integrate syntactic and statistical information, the thesis makes a case for inferring information from very small training corpora, and provides methods to do so.

Contents

Acknowledgments

During my work on this thesis, I received a lot of support from my thesis committee, and I would like to acknowledge their help. First, I would like to thank my committee chair Alon Lavie, who carried me through a number of tough phases and believed in me and the project. I thank Jaime Carbonell for his insightful advice. I, like many others, found every meeting with him a step forward. I thank Lori Levin for her linguistic perspective and input that helped form this thesis. I also want to thank her especially for all the work she did in helping me improve my thesis document. Last but not least, I want to thank Bonnie Dorr for her input. She gave very thorough comments which showed her decication; for this I am grateful, and her advice made my thesis better.

I would also like to thank all the faculty, students and staff of the AVENUE project under which this research was performed, and its international partners. In particular, thanks to Erik Peterson for the transfer engine, which I used extensively in this work, and without which my work could not have happened. Alison Alvarez deserves thanks for creating the copula elicitation corpus. Ariadna Font-Llitjos provided very useful technical comments and indispensable moral support. I would also like to thank our numerous Hindi and Hebrew informants, who helped especially with data collection and morphology.

Although this work was done at Carnegie Mellon University, I give special thanks to Linda Lankewicz, my undergraduate advisor and mentor at the University of the South, who taught me a lot about computer science, research, and life, and advised me to go to graduate school.

Guy Lebanon deserves my infinite gratitude for never, ever doubting me, for giving me moral support whenever I needed it, and for playing a major role in making my life as a PhD student happy.

My family, Maria and Lorenz Probst as well as Barbara Probst-Jilg and Martina Bug, always thought I could do it, and saw me through it to the end. I am thankful for their support, and thankful to my parents for being

proud of me.

This research was funded in part by the DARPA TIDES program and by NSF grant number IIS-0121-631.

Chapter 1

Introduction

1.1 Motivation

Traditionally, it was common to spend many years of effort on the development of machine translations (MT) systems for a given language pair. Recent advances in MT research have focused on more speedy development of systems. The goal is to be able to apply MT to a new language pair within a matter of weeks or months. This is clearly a desirable goal, because so far MT has only been applied to a select few language pairs. With the advance of technology all over the world, bridging language barriers is becoming ever more important, and MT can be a very useful tool in this context. In practice, however, this multi-lingual challenge requires an effort by the MT community to develop techniques that allow the rapid development and deployment of MT systems. Clearly, human involvement must be minimized if a system is required within a matter of months.

The multi-lingual challenge is one of the reasons that many recent approaches to MT have opted to learn translation information automatically from bilingual corpora (i.e. text that is given in parallel in two languages), most commonly using example-based or statistical methods that derive models of translation. Statistical and example-based approaches focus on analyzing the *sequence* of words and their translation. For example, how does a given sequence of words translate into another language, and (in the case of statistical MT) with what probability? Statistical and example-based MT approaches have the advantage of being derivable from bilingual text, thus overcoming the development bottleneck. However more recently, the community has noted once again the potential beneficial impact of structural and feature information on translation quality. Structural information can

be in the form of constituent transfer: for example, how is a noun phrase or a sentence constructed in a language, and how does the ordering of words and groups of words change when translated into another language? In such approaches, the composition and hierarchical organization a sentence is analyzed, and rules are given for how the hierarchy transfers into another language. For example, the English sentence 'THE CAT SLEPT' is considered to consist of two constituents, a noun phrase 'THE CAT' and 'SLEPT'. Structural transfer information would then address such questions as whether the noun phrase and verb phrase appear in the same order in another language, whether the noun phrase appears in the same composition, a determiner 'THE' followed by a noun 'CAT', and if not, how this noun phrase is expressed, etc. Feature information addresses another set of problems. For example, the noun phrase 'THE CAT' is considered to be singular, as there is only one 'CAT'. Feature information then addresses such problems as how singular is expressed in the other language, and how it can be assured that the MT system will produce the equivalent of one 'CAT' rather than of many 'CATS'. Naturally, things become quite more complicated when the transfer information is to be used on sentences of 30 or 50 words with complicated structural composition. Despite such complications, structural and feature information can be very useful for translation: it allows the grouping of words into meaningful elements (such as noun phrases), that can then be translated as a whole. It further allows the system to capture more information about the sentence, for example whether a noun phrase is singular. Such information can be used to produce the correct translation.

Most statistical and example-based systems do not infer structural or feature information. This thesis, on the other hand, ultimately aims at making use of such structural information by learning syntactic transfer rules, i.e. transfers between structures in one language and structures in another language. The transfer rules are learned from a very small bilingual corpus. In particular, this work specifically targets scenarios where bilingual data resources are extremely limited, and where the target language is a major language such as English, and the other language is a minor language. We define a minor/major language pair simply as a pair of languages where 1) little or no parallel data is available, and where 2) there is no syntactic parser for the minor language. Parallel data is data that is given in one language with the translation of each sentence or phrase in the other language. A syntactic parser is a tool that gives the structural composition of a sentence in the form of a tree.

Our definition of minor/major does not necessarily imply that there are no other resources (such as monolingual data, morphology modules,

etc.) available for the minor language. This can in fact be the case, and in particular a morphological analyzer for the minor language is put to effective use in our approach. In addition, we also make use of translation lexicons that specify the translation of individual words or short sequences of words, and also contain information about the part of speech (e.g. noun, adjective, verb, etc.) of the translated words. The experiments reported in this thesis use Hebrew and Hindi as minor languages. Furthermore, the transfer rules that are learned are directed: run-time translation is always done from Hebrew or Hindi into English.

The learned grammar rules consist of both source- and target-language information where the source language (SL) is the language translated from and the target language (TL) is the language translated into. The rules contain a *context-free part* that captures structural information and an optional set of *unification constraints* that capture feature information. When the rules apply to unseen test data, they provide a means to analyze the SL sentence or phrase and map its structure into a structure that is appropriate for the TL. The run-time system that uses the learned rules to produce translations is a statistical transfer system consisting of a transfer engine that produces partial translations, and a decoder that finds the most likely combination of the partial translations. Thus, the learned rules are used in the context of a larger translation system.

The algorithms laid out in this thesis infer syntactic transfer rules from bilingual, word-aligned text. The rules are learned in three stages. In the first, we create hypothesis rules, so-called *seed* rules, for each training example. The seed rules can only apply in isolation, and reflect the training examples very closely. The following learning step, Compositionality, aims at generalizing from the training data by capturing higher-level structural information, resulting in rules that can combine with each other. In the third learning step, we add grammatical constraints to the rules, thereby restricting when a rule can apply, or restricting what translations a rule can produce.

A major constraint imposed on this work, and a major difference between this and related projects, is that the work described here does not assume the availability of a parser for the minor language. Due to the absence of a parser for one language, we frame the learning problem as the task of learning 1) the structure of the language for which there is no parser, and 2) the transfers of structures between the two languages. More specifically, the rule learner must make use of a *target-language parser at training-time*, but must infer rules that can be used for *source language parsing at run-time*. This is a potentially confusing point: the training system (this thesis) has

available to it parses for the target-language side of each bilingual training example. In our language pairs, this means that we have the parses for all English sentences in the training examples. At run-time, we do *not* have the parses for the source language, in our case Hebrew or Hindi. Instead, during training we must infer rules that can, at run-time, parse the Hebrew or Hindi input. This inference must happen using only the English parse. At run-time, the English parse of the training examples is not used, as we are translating *into* English. This is a challenging problem, because it essentially involves inferring the grammar of one language using parses for another language.

The other major difference between this and related work is that because of our interest in minor languages, we are forced to learn from extremely small bilingual datasets. In fact, most of the grammars described in this thesis were learned from a bilingual training set of 120 sentences and phrases.

One big challenge of this thesis is to learn transfer rules using a variety of imperfect and sometimes inconsistent resources, and do so fast, i.e. without large amounts of language-specific engineering. Most notably, we use an English syntactic parser, morphology modules for both languages, a bilingual dictionary (i.e. a translation lexicon), and a small bilingual corpus. One of the challenges of this work is to integrate the different knowledge sources meaningfully, especially when they contradict each other.

This document is structured as follows: we will first state the research goals and give the thesis statement as well as a thesis statement. We will then describe related work in the area. This is followed by a discussion of the training and run-time systems: what is the input of the rule learning system, what is the output, and how is this output used to produce translations at run-time? We will then discuss each of the three rule learning phases. The following chapter presents an evaluation of the learned rules on Hebrew→English and Hindi→English translation tasks. We conclude with lessons learned, contributions of this thesis to the MT community, and a discussion of possible future work.

1.2 Research Goals

In this section, we summarize the goals of the presented research. The premise of the thesis is to automatically learn transfer rules for Machine Translation from bilingual text. More specifically, we target a difficult scenario where rules are learned from 1) a very small corpus and 2) for an unbalanced language pair where a parser is available for only one of the

languages. The learned rules are to be used in a run-time system that combines transfer and statistical techniques. In the following, we discuss each of our research goals. At the end of the document, we will return to these goals, summarize how they were accomplished, and state in what way they constitute novel research contributions.

1. **Develop a framework for learning transfer rules from bilingual data.** The first goal of the thesis work is to develop a novel framework for learning syntactic transfer rules. Many groups have inferred structural or dependency information from text, but our approaches is novel in that the rules include 1) a context-free backbone and 2) unification constraints. Learning both of these enables us to capture a wide variety of linguistic phenomena.

2. **Improve of the quality of MT output by automatically learned rules.** The inferred rules are intended to be used in an MT system in order to improve translation quality. The system is a hybrid transfer and statistical system, where the transfer rules are used to produce a possibly large number of hypothesis translations, and a statistical selection module is used to choose between the hypotheses. The rule learning algorithms are to be designed such the rules can be used in this hybrid system effectively.

3. **Address limited-data scenarios with 'frugal' techniques.** This thesis is aimed at scenarios where bilingual data is in sparse supply or not available. In order to overcome the lack of a large bilingual corpus, the rules are learned from a carefully designed, but extremely small corpus that is translated and word-aligned by a bilingual user, meaning that the user specifies which words translate into each other. This small corpus serves as the only training data for the rule learning system. The goal is to demonstrate that meaningful transfer rules can be learned from very small datasets, therefore making a case for MT under frugal data scenarios.

4. **Learn rules in the absence of a parser for one of the languages.** The language pairs that are targeted in this work are 'unbalanced', meaning that a syntactic parser is available for one of the languages in the pair (the major language, in our system the target language), but not for the other (the minor language, in our system the source language). The learning algorithms must be designed to overcome the lack of syntactic information for the minor language, and must learn such information automatically.

5. **Combine a set of different knowledge sources in a meaningful way, and do so quickly.** The premise of this approach is to combine as many existing knowledge sources as possible. The following resources are used in the approach:
   Required resources: 1) small word-aligned bilingual corpus, 2) parser for major language
   Optional resources: 1) morphology module for major language, 2) morphology module for minor language, 3) bilingual dictionary

   One of the challenges of this thesis is to combine these knowledge sources meaningfully and quickly, i.e. without extensive language-specific engineering, and to produce rules that can in fact improve translation quality. This is challenging because all of the resources are incomplete, they achieve good but not perfect accuracy, and they sometimes disagree.

   Note that we list here the resources required for the *learning* phase, i.e. the phase where the transfer rules are inferred. A different set of resources is necessary for the run-time MT system in which the learned rules are embedded. The run-time systems will be discussed in section 1.4.2.

6. **Encouraging MT research in the direction of incorporating syntax into statistics-based systems.** There has recently been increased interest in combining syntactic and statistical information for Machine Translation. This thesis aims at demonstrating that this combination can be done successfully as well as automatically. We envision that the combination of different approaches will become the standard accepted approach in the future, and our goal is to encourage this trend.

7. **Learn human-readable rules that can be improved by an expert.** The automatically learned rules are in a format that is human-readable by expert users. This has two big advantages. The first advantage is the rules can be proof-read and refined by a computational linguist who knows both languages, or by a rule refinement system that analyzes the behavior of the rules and corrects them based on the output they produce. (Font-Llitjós, 2004) has developed algorithms for this task, and has simulated automatic rule refinement manually, thus showing that both manual and automatic rule correction are feasible. The second advantage is that the automatically learned rules can easily be used in conjunction with manually written rules that

follow the same formalism. This allows for maximum flexibility: our system uses produces a set of rules that could be refined or expanded by an expert user.

1.3 Thesis Statement

The thesis statement is given as follows:

1. Given bilingual, word-aligned data, and given a parser for *one* of the languages in the translation pair, we can **learn a set of syntactic transfer rules for Machine Translation**. The rules are comprehensive in the sense that they include analysis, transfer, and generation information.

2. The rules are learned from a **small** bilingual corpus, and the learning algorithms are optimized for this task.

3. The rules consist of a **context-free backbone and unification constraints**, learned in three separate stages.

4. The resulting rules form a syntactic translation grammar for the language pair and are **used in a statistical transfer system to translate unseen examples**.

5. **The translation quality** of a run-time system that uses the learned rules **is superior to a system that does not use the learned rules** on Hebrew→English and Hindi→English translation tasks.

6. **The translation quality is comparable to the performance using a small manual grammar written by an expert** on the Hebrew→English translation task.

7. The thesis presents a **new approach to learning transfer rules for Machine Translation** in that the system learns syntactic models from text in a novel way, aiming at **producing rules of the same type that a human grammar writer would design**.

1.4 Thesis Summary

In this section, we give a summary of the thesis, presenting the rule formalism, the MT system within which the learned rules are embedded, and a brief outline of the learning algorithms and evaluation results.

1.4.1 Transfer Rule Formalism

In order to discuss the learning algorithms, we must first define the goal, i.e. what kinds of transfer rules we wish to infer. An example of a Hebrew→English rule looks as follows, presented here with a possible parallel sentence pair that this rule could translate.

```
Parallel sentence pair:
Hebrew: H   &NIN     H   RB          B      BXIRWT
Gloss:  the interest the widespread in+the election
English: the widespread interest in the election

Transfer rule:
NP::NP ["H" N "H" ADJ PP] -> ["THE" ADJ N PP]
(
(X2::Y3)
(X4::Y2)
(X5::Y4)
((Y3 NUM) = (X2 NUM))
)
```

Rules consist of a context-free part and possibly a set of unification constraints. The first context-free component of every transfer rule is the source and target language type information, 'NP' in the sample rule above. The type information is followed by a part-of-speech or, more generally, component sequence, both for the source and the target languages. In the example rule, the component sequences are ["H" N "H" ADJ PP] for the SL and ["THE" ADJ N PP] for the TL. Entities enclosed in quotes are actual lexical items, whereas entities that are not enclosed in quotes are parts of speech (i.e. a N can be filled by any noun) or else generalized constituents (e.g., PP is a prepositional phrase). At run-time, the SL component sequence is used to parse an input sentence, both to build a SL parse tree, and to check if the rule applies to the input sentence. The TL component sequence is used to build the TL tree that corresponds to the SL parse tree. This is done by transferring the SL tree into the TL using the alignments, e.g., (X2::Y3) above, indicating the that second SL component ('&NIN') aligns to the third TL component ('INTEREST'). "X0" is reserved for the top-level, i.e. the root node "NP", on the source language side, and "Y0" is reserved for the target language top level, also "NP" here.

All rule parts described so far are context-free, and all are required. Unification constraints, on the other hand, are optional. The above rule

has one unification constraint, '((Y3 NUM) = (X2 NUM))'. Constraints are used as specification information, specifying to what sentences a transfer rule applies, what types of features are required when generating the target language sentence, and what feature values transfer from the source into the target language.

One way to understand the rules is that they encode tree-to-tree transfers at different levels. Each individual rule captures a small tree-to-tree transfer. When the rules are combined, the x-side type and component sequences combine to a large tree which is then mapped to an equivalent tree on the y-side, which is again built from smaller trees from individual rules.

1.4.2 Run-Time System: Transfer Engine and Decoder

The transfer engine is a system that takes as its input a translation lexicon, a set of rules of the formalism described above, morphological information for one or both languages if available, and source language input text, and it creates target language output. This tool was developed by Erik Peterson (Peterson, 2002).

The transfer engine uses the rules to analyze the source language sentence. It first builds the set of all those parse trees that are licensed by both the lexicon and the grammar rules (if there are any). The resulting set is then transferred into the target language, applying constraints and reorderings as specified in the rules. The final product is a (generally large) 'lattice' that contains *all* possible partial translations for a given input sentence. The individual lattice entries, i.e. partial translations, are referred to as 'arcs'. Arcs are indexed by the portion of the input sentence that they span. Individual arcs rarely cover the entire input sentence. Rather, the lattice contains all possible partial translations of all input chunks of length 1 up to the sentence length. Note that 'all possible translations' is defined as all translations that are licensed by the grammar and lexicon. The rules are applied to the input sequence, and not to all permutations of input words, i.e. the rules must apply to input word sequences in the order in which the input words are given. The design and implementation of the transfer engine is not in the scope of this thesis, but is described in (Peterson, 2002).

After the transfer engine produces a list of all possible partial translations, the lattice is processed by a statistical decoder. This tool is the statistical decoder that is used in the in-house statistical MT system. For details, see (Vogel et al., 2003).

The decoder uses an English (or other major language) language model in order to produce the most likely full translation. It does this by performing

a Viterbi search through the lattice, selecting those arcs whose combination will result in the full translation with the highest probability. For more details on the statistical decoder, as well as the entire run-time system, please refer to (Lavie et al., 2003).

1.4.3 Training Data

The rule learning training corpus that is used for the experiments described in this thesis is a small but carefully designed corpus that is aimed at covering a wide variety of structural phenomena. It consists of 120 sentences and phrases, and it covers different compositions, i.e. examples of different sequences of parse child nodes, of adjective phrases (ADJPs), adverb phrases (ADVPs), noun phrases (NPs), prepositional phrases (PPs), SBARs and sentences (Ss). These types include a small number of subtypes such as SQ (a question sentence).

For illustration, we give here several examples of different compositions of ADJPs. We would have examples of the following composition (among others):

```
ADJP -> ADJ
ADJP -> ADV ADJ
ADJP -> ADV ADJ PP
etc.
```

For each of the types of structures, we have collected a number of instances of different composition. In order to obtain good coverage of the most frequent compositions, we analyzed the frequency of compositions that were observed in the Penn Treebank (Marcus et al., 1995), and found examples for the most frequent structures.

The structural elicitation corpus consists of 120 sentences and phrases in English. The bilingual data is obtained via a process of elicitation, where a bilingual user translates the data from English into their native language and specifies the word alignments. Word alignments are translations of individual words or phrases into each other. The structural corpus was translated into both Hebrew and Hindi. Unless otherwise noted, all learned grammars that are described in this document were learned from this corpus, which is referred to as the 'structural corpus'.

As was said above, the training algorithm makes use of the English parse of each parallel training example. For this reason, we include the parses in the training data. Some examples of training examples can be found below,

where we present the bilingual sentence pair together with the user-specified word alignments and the English parses.

```
English: in the forest
Hebrew: B H I&R
Alignment: ((1,1),(2,2),(3,3))
C-Structure:(<PP> (PREP in-1)(<NP> (DET the-2)(N forest-3)))

English: quickly
TL: B MHIRWT
Alignment: ((1,1),(1,2))
C-Structure:(<ADVP> (ADV quickly-1))

English: the boy ate the apple
Hebrew: H ILD AKL AT H TPWX
Alignment: ((1,1),(2,2),(3,3),(4,5),(5,6))
C-Structure:(<S> (<NP> (DET the-1)(N boy-2))(<VP> (V ate-3)
    (<NP> (DET the-4)(N apple-5))))

English: a dispute with the school board
Hebrew: SKSWK &M W&D BIT H SPR
Alignment: ((2,1),(3,2),(4,5),(5,4),(5,6),(6,3))
C-Structure:(<NP> (<NP> (DET a-1)(N dispute-2))
    (<PP> (PREP with-3)(<NP> (DET the-4)(N school-5)
    (N board-6))))

English: old
Hebrew: I$N
Alignment: ((1,1))
C-Structure:(<ADJP> (ADJ old-1))

English: where our interests lie and what we must do
Hebrew: HIKN $ H AIN@RSIM $LNW NMCAIM W MH $ ANXNW CRIKIM
    L&$WT
Alignment: ((1,1),(2,5),(3,4),(4,6),(5,7),(6,8),(7,10),(8,11),
    (9,12))
C-Structure:(<SBAR> (<SBAR> (<WHADVP> (WH where-1))
    (<S> (<NP> (POSS our-2)(N interests-3))(<VP> (V lie-4))))
    (CONJ and-5)(<SBAR> (<WHNP> (SUBORD what-6))
    (<S> (<NP> (PRO we-7))(<AUX> (AUX must-8))
```

```
    (<VP (V do-9)))))

English: aware that the money was gone
Hebrew: MWD& $ H KSP N&LM
Alignment: ((1,1),(2,2),(3,3),(4,4),(5,5),(6,5))
C-Structure:(<ADJP> (ADJ aware-1)(<SBAR> (SUBORD that-2)
    (<S> (<NP> (DET the-3)(N money-4))(<AUX> (V was-5))
    (<ADJP> (ADJ gone-6)))))

English: because he was hungry
Hebrew: KI HWA HIH R&B
Alignment: ((1,1),(2,2),(3,3),(4,4))
C-Structure:(<SBAR> (PREP because-1)(<S> (<NP> (PRO he-2))
    (<VP> (V was-3)(<ADJP> (ADJ hungry-4)))))
```

1.4.4 Seed Generation

Seed Generation is the first of three learning phases. It takes as input the training corpus described in the previous section, uses a variety of resources such as the parser for the major language (TL), and produces an approximation of learned transfer rules. For each training example, it constructs an initial transfer rule called a *seed rule*. The seed rule is a complete transfer rule in format: it contains SL and TL type information, component sequences, and alignments. In other words, it is a fully functional transfer rule, transferring an SL structure into a TL structure, where the minor language (Hebrew/Hindi) is the source language and the major language (English) is the target language.

For each training example, the task of the Seed Generation module is to produce the following rule parts: SL type information, TL type information, SL component sequence, TL component sequence, and word-level alignments.

The following is an example of a flat rule:

```
Parallel sentence pair:
Hebrew: H   &NIN     H   RB         B      BXIRWT
Gloss:  the interest the widespread in+the election
English: the widespread interest in the election

Transfer rule:
NP::NP ["H" N "H" ADJ PREP DET N] -> ["THE" ADJ N PREP DET N]
```

```
(
(X2::Y3)
(X4::Y2)
(X5::Y4)
(X6::Y5)
(X7::Y6)
)
```

1.4.5 Structural Learning

The second learning phase aims at inferring rules of higher-level structures. Wherever possible, the rules learned in this section capture transfer between constituents such as noun phrases, prepositional phrases, etc. When generalization to the constituent level is impossible or undesired, the rules transfer POS sequences and/or lexical items. Thus, the structural rules provide a mixture of transfer at different levels, depending on what can safely be inferred from the training data.

Before designing the learning algorithms, we must specify what types of structures can and should be learned from the data. Otherwise, the learning algorithm could infer rules that are either faulty or cannot be used by the run-time system. In chapter 7, we discuss in detail the types of possible and meaningful transfers.

The basic idea of structural learning is to infer *compositional* rules, so that the rules can combine with each other and cover a wide variety of phenomena at run-time. This is achieved by traversing the English parse from the top-down, and introducing a compositional element for a subnode wherever appropriate. Two basic settings are possible for Compositionality Learning: assuming Maximum Compositionality, or not assuming Maximum Compositionality, described in chapter 7.

In both settings, the goal of Compositionality is to generalize as much as possible over the training data and thus produce rules that will apply to a variety of new contexts. By contrast, Unification Constraint Learning aims at limiting the applicability and/or the output of the rules. In this way, Compositionality and Constraint Learning together serve to strike a balance between generality and overgeneralization that is supported by the training data.

An example of a compositional rule can be seen below:

```
Parallel sentence pair:
Hebrew: H   &NIN     H   RB          B      BXIRWT
```

```
Gloss:  the interest the widespread in+the election
English: the widespread interest in the election

Transfer rule:
NP::NP ["H" N "H" ADJ PP] -> ["THE" ADJ N PP]
(
(X2::Y3)
(X4::Y2)
(X5::Y4)
)
```

1.4.6 Learning Unification Constraints

The third learning phase adds unification constraints to the context-free rules produced in the previous two phases. Unification constraints can either 1) limit their applicability to certain contexts (thereby limiting parsing ambiguity), 2) ensure the passing of a feature value from source to target language (thereby limiting transfer ambiguity), or 3) disallow certain target language outputs (thereby limiting generation ambiguity). As in the previous chapter, the constraints are automatically learned from data. Constraints can be subdivided into *value constraints* with specific feature values (e.g., ((X1 NUM) = S)) and *agreement constraints* that enforce agreement between features (e.g., ((X1 NUM) = (X4 NUM))). A taxonomy of possible and relevant constraints can be found in section 8.3.

Constraint Learning can again be subdivided into three main phases. In the first phase, basic constraints, mostly in the form of value constraints, are introduced using the morphology modules. In the second phase, the basic constraints are generalized to agreement constraints wherever appropriate. Finally, most of the basic constraints are eliminated, unless they can serve to disambiguate between structures during transfer.

The basic constraints gather as much information as is available from the morphology modules and introduce a large set of value constraints. The goal of the basic constraint phase is to obtain all morphological features for each word and mark it in the transfer rule as a value constraint. Such information can be obtained from the morphology modules. This module also determines what features should be marked on the constituent level, not only on the word level. For example, NPs should be marked for number and definiteness. This is important when rules are used in combination with each other. Some rules may only be valid when used with a singular NP, for example.

Agreement constraints are learned from the value constraints that were produced as basic constraints. This is achieved again in three distinct phases. The first two phases focus on intra-lingual constraints, i.e. SL constraints and TL constraints. Since one language does not influence what agrees in the other language (e.g., Hebrew has no influence on subject-verb agreement in English), these constraints can be learned separately. Value constraints are generalized to agreement constraints if it is found that two values should always agree. For example, if two components such as X1 and X4 are generally marked with the same number value, then an agreement constraint should enforce that the rule applies only if this is the case. The decision about whether an agreement constraint should be introduced can be made by considering the full training set, and collecting statistics of how often components agree in a certain feature value, and how often they do not. We employ two types of tests, depending on how many data points are available for a specific potential agreement constraint. The first test is a likelihood ratio test, and the second is a heuristic fallback. All agreement constraints are learned in this fashion.

The last Constraint Learning phase determines which of the value constraints should be retained and which ones unnecessarily limit the applicability of the rules. We retain only those value constraints that can be used to disambiguate: in some cases, a specific feature value determines what TL component sequence should be produced. This can be tested automatically by finding examples where a specific value resulted in a specific TL structure, whereas a different value produced a different TL structure.

Below we give an example of a rule annotated with a unification constraint:

```
Parallel sentence pair:
Hebrew: H   &NIN     H   RB         B      BXIRWT
Gloss:  the interest the widespread in+the election
English: the widespread interest in the election

Transfer rule:
NP::NP ["H" N "H" ADJ PP] -> ["THE" ADJ N PP]
(
(X2::Y3)
(X4::Y2)
(X5::Y4)
((Y3 NUM) = (X2 NUM))
)
```

1.4.7 Summary of Results

In this section, we present very abbreviated evaluation results. We evaluated our learning algorithms most extensively on a Hebrew→English evaluation task. While in section 9 we present different sets of results for different corpora, different system settings, as well as for Hindi→English translation, we will present here only briefly evaluation results for the default settings. We use the standard automated evaluation metric BLEU (Papineni et al., 1998), as well as a slightly modified version, ModBLEU (Zhang & Vogel, 2004), and an evaluation metric called METEOR (Lavie et al., 2004). Note that none of these metrics is normalized between 0 and 1, so that what is crucial is the relative difference in score between the systems, not the absolute scores. The evaluation was run on an unseen test set of 62 sentences of newspaper text with two reference translations (test set 2). We compare the run-time system in three settings: one without any grammar rules, i.e. a statistical-only system, one with the learned grammar rules, and one with a small manually written grammar. The results can be found in Table 1.1.

Grammar	**BLEU**	**ModBLEU**	**METEOR**
No Grammar	0.0565	0.1362	0.3019
Manual Grammar	0.0817	0.1546	0.3241
Learned Grammar	0.0780	0.1524	0.3293

Table 1.1: Evaluation results for Hebrew→English translation on test set 2 of 62 sentences of newspaper text.

It can be seen from the results that our approach results in significant gains in translation performance. The relative improvement depends in part on what evaluation metric is used. However, for all three evaluation metrics the performance gain is considerable as well as statistically significant. It can also be observed that the METEOR score assesses the translation quality of the fully automatic system as higher than with the manually written grammar. This is a very positive result for our approach.

Chapter 2

Related Work

In this chapter, we discuss related work and situate this thesis within the Machine Translation field. In particular, we discuss several approaches to MT and how they relate to the present thesis. We discuss the advantages and disadvantages of previous approaches, and compare them to the strengths and weaknesses of the approach presented here.

We will further compare different approaches to inferring structural information from monolingual or bilingual text. The automatic inference of generalized information has received a fair amount of attention especially in the example-based and statistical MT frameworks. Remarkably little has been done on the inference of complex rules within a transfer-based system. It should however be noted that the boundaries between the classical approaches to MT become blurred under all the paradigms discussed here. For example, a statistical system that incorporates context-free transfers is essentially a hybrid between a transfer and a statistical system. Similarly, our system is a hybrid transfer-statistical approach, as will be described in chapter 3 below.

Finally in this chapter, we will address the task of building MT systems from small datasets. The approach described in this thesis learns from a very small, but carefully developed corpus. Similar approaches have been taken by a few other research groups. The idea of a carefully constructed corpus is especially attractive when building MT systems for language pairs where little bilingual data is available to begin with.

It is not straightforward to situate our work within the larger MT community. This is because a large number of papers describe work that is related to our approach in some way. To our knowledge, however, nobody has addressed the particular problem that we are addressing in this thesis,

and under the circumstances we propose. For instance, some systems learn transfer rules for transfer-based MT, but they generally use parses in both languages. Other systems only use a parser for one language, but they only allow for a small number of reorderings and do not encode linguistic features.

2.1 Approaches to Machine Translation

2.1.1 Transfer-based Approaches to MT

Especially in the 1980s and early 1990s, MT systems have relied heavily on hand-written rules: transfer rules that specify how a (syntactic or semantic) structure in one language maps to the corresponding structure in another language. Under this paradigm, the translation task is split into three distinct phases: 1) analysis of the SL sentence, which generally includes syntactic and morphological analysis, 2) transfer into a TL representation, and 3) generation of a TL sentence, again using syntactic and morphological information. The knowledge is stored in two separate resources, the grammar (i.e. the set of transfer rules) and the lexicon. Both of these resources together serve to accomplish the three steps of translation. The lexicon and grammar store varying amounts of information, and how much information is stored in each depends on the specific system design.

A classical example in this context is Systran (Hutchins & Somers, 1992; Senellart et al., 2001). Other systems that have received a large amount of attention include the METAL system (McCormick, 1998; Thurmair, 1990), the GETA-Ariane system (Boitet, 1988), and finally the Eurotra system (Alberto Alonso, 1990; Steiner, 1990).

The system within which this thesis was developed fits best into the framework of transfer-based MT. The Avenue system, as described below, also employs a separate lexicon and a grammar that encodes transfers between source and target language. There are a number of differences between our system and previous systems, however. The first is that in the Avenue system transfer is combined with a statistical decoder, resulting in a hybrid MT system. Hybrid MT systems have been addressed by a minority of researchers and are described in section 2.1.3. The second difference is that unlike in the classical approach, the transfer rules, i.e. the grammar, in the Avenue system are learned automatically. This is the goal of this thesis.

The approach described here can thus be summarized as follows: it aims at overcoming the development bottleneck for transfer-based MT in the context of a hybrid transfer-statistical system.

2.1.2 Statistical and Example-Based MT

While the classical transfer-based approach can achieve reasonably high translation quality, it also takes many person years to create a system that can handle a wide variety of sentence constructions and will perform well on unseen text. One approach to overcoming the development bottleneck is the one described in this document: learning grammar rules automatically from bilingual text.

Other approaches, which currently receive considerable treatment in the research community, are corpus-based approaches, namely statistical MT (SMT) and example-based MT (EBMT). Much attention has recently been given to the rapid deployment of MT systems: given a language pair, how quickly can we ramp up a system that can produce reasonable quality output?

EBMT builds a database of word- and phrase-level associations (e.g., (Sato & Nagao, 1990),(Brown, 1997)) from a bilingual corpus. The approach is based on the observation that the bilingual corpus is essentially a database of correct translations. If the system can determine which parts of the SL sentences translate into which parts of the TL sentences, then it can build a repository of known correct translations. At run-time, a SL sentence is matched as a whole or in part against the translation examples, and is translated using those examples. The partial translations are then merged to form a full translation of the SL sentence.

SMT is similar in spirit: it builds a 'memory' of word and phrase translations from a bilingual corpus. The approach is however very different. Most SMT systems are built based on a noisy channel model that allows the translation problem to be decomposed into 1) a translation modeling problem and 2) a target language modeling problem (e.g., (Vogel et al., 2003), (Brown et al., 1993), (Och & Ney, 2002)). The bilingual corpus is used to derive translation probabilities between words and phrases, and (possibly in conjunction with a monolingual TL corpus) to derive a language model of the target language. At run-time, the translation model is used to build a list of possible partial translations, and then the target language model is used to choose between the partial translations and to merge them into a full translation. Because the problem was framed as a noisy-channel model, the run-time SMT system is generally referred to as a 'decoder'.

EBMT and SMT are indirectly related to the work presented in this thesis: In the Avenue system, we employ a statistical decoder much like in an SMT system to disambiguate between partial translations (described in greater detail in chapter 3). The statistical decoder takes in a list of

possible partial translations. It then optionally reranks them according to a translation model that is learned from a bilingual corpus. Finally, it finds the most likely combination of partial translations, based on an English language model, and merges them to a full translation. In this sense, the AVENUE hybrid system is a hybrid system between a transfer-based and a statistical system.

Further, both EBMT and SMT are relevant to the present work because many researchers have addressed the problem of incorporating syntactic or other generalized structural information within SMT or EBMT systems. Such approaches blur the boundaries between SMT, EBMT, and transfer-based approaches, and many of the systems that identify themselves as SMT or EBMT papers are nevertheless relevant to this work. These approaches will be described in 2.2.

2.1.3 Hybrid Approaches to MT

As already mentioned briefly above, the approach described here learns transfer rules that are used in the AVENUE system, a hybrid MT system that combines syntactic and statistical information.

Hybrid MT is generally based on the idea that symbolic (i.e. syntactic, morphological, etc.) information can be very useful for capturing and transferring linguistic information, but that those methods often result in a large number of ambiguities. For this reason, the systems create a large number of (partial) hypothesis translations using symbolic methods, and then employ a statistical model to extract the best hypotheses and to form a full translation. This is the case in the system described in (Knight et al., 1995; Knight et al., 1996), where the system first produces a syntactic representation of the sentence, which is then mapped to a semantic representation, where ambiguities are preserved. The semantic representations are then ranked using a statistical model. A symbolic decoder then produces candidate translations, which are again reranked and filtered with a statistical model, more specifically a target language model.

An approach that is related to the one presented here, 'Generation-Heavy MT' (Habash & Dorr, 2002; Habash, 2002; Ayan et al., 2004) passes a rich set of translation hypotheses to a language model in the target language, which then extracts the best combination of hypotheses. The approach targets in particular language pairs that differ greatly in structure as well as in the availability of syntactic and semantic tools. SL analysis is done with a dependency parser. Then the lexical items on the dependency tree are translated using a translation lexicon. This is followed by a series of TL steps

that deal with categorial variation, subcategorization variation, structural expansion (such as head swapping) to handle translation divergences, etc. These steps are hand-coded by humans. A TL language model then picks between competing partial translations to form a full translation.

In this thesis, we describe an approach to learning transfer rules that are used in a hybrid MT system. The design of the hybrid MT system is not directly part of this thesis. However, it is important to discuss hybrid approaches here: the transfer rules that are learned automatically often overgeneralize in order to cover a wide variety of structures. The fact that a statistical decoder is charged with reranking and filtering the hypothesis translations affects the design of the learning algorithms described in this thesis. Seed Generation and Compositionality are aimed specifically at generalizing from the training data as much as possible (and plausible). This decision fits with the overall system design of a symbolic, overgeneralizing, module and a statistical module that chooses between hypothesis translations.

2.2 Automatic Inference of Generalized Structure

2.2.1 Monolingual Grammar Induction

When framing the problem of grammar induction very generally, statistical parsing techniques should be mentioned. Most modern statistical parsing techniques view the problem of parsing as a supervised learning problem (Charniak, 2000), (Collins, 2003). Here, implicit grammars are learned from a large corpus of parsed data, usually the Penn Treebank (Marcus et al., 1995).

Some projects have focused on less supervised methods. (Hwa, 1999) describes an approach to grammar induction from less supervised data by pre-selecting a smaller corpus to label. An unsupervised approach starting from a POS tagged, but not parsed, corpus is described in (Klein & Manning, 2001). Here, a context-free grammar is induced based not on a hand-parsed corpus, but rather on linguistic principles of constituency. A POS sequence is described in terms of the tags that surround it, how many different such contexts there are, how frequent they are, etc. This allows not only for the detection of salient POS patterns (by the entropy of the contexts it appears in), but also for the definition of a similarity metric between sequences. Then a context-free grammar can be learned by clustering techniques. The learned grammars, as expected, do not perform as well on parsing as models trained in a supervised fashion. They do however capture linguistically

viable rules in an unsupervised fashion. Like most unsupervised, statistics-based techniques, this technique requires a larger training corpus than the one we use in the approach described here.

The work described in this thesis can be viewed as falling within the semi-supervised methods. This is because we assume the existence of a parser for the major language. For the minor language, however, we only assume that we can reasonably reliably obtain parts of speech and morphological information.

2.2.2 Inference of Structural Rules for Transfer-based MT systems

A few researchers have proposed approaches to learning structural information from bilingual data within the framework of transfer-based systems. Generally, those systems differ from ours in that they infer transfers by parsing both the source and the target language sentences at training time. Such algorithms are designed to extract generalized transfers from the aligned trees.

It should be noted that some of the systems described in this section could also be considered generalized EBMT systems (cf. section 2.2.3). As was said above, the boundary between transfer-based MT and example-based MT becomes blurred, especially when the transfers are (partially) lexicalized, and when they are learned automatically from bilingual text. However, the systems described in this section fit also into the transfer paradigm, in particular because they use full parses as training data.

(Menezes & Richardson, 2001) use such an approach: pairs of aligned sentences in 'Logical Form' (LF) serve as input to the training algorithms. The LF is essentially a dependency structure that is tagged with morphological information. The task of the training algorithm is then to 1) find correct alignments between the logical forms, and 2) break the full logical forms into smaller parts that can be used as transfer rules. Alignments between sentence parts are allowed only if they meet linguistically motivated criteria. For example, a V+Object combination can be aligned to a verb, but not to an NP. The extracted rules are at least partially lexicalized; each rule must contain at least one lexical item on each language side. The transfer rules are generalizations in that they contain variables such as parts of speech or constituent labels that can be filled by other rules.

Another approach that uses parse trees for both languages as training data is described in (Meyers et al., 1998). This system also grows alignments between parsed sentence pairs, where the parser of choice is a dependency

parser. The authors limit the search space by a linguistically motivated heuristic: in their system, no dominance-violating alignments are allowed. For example, if a_{SL} aligns to a_{TL} and b_{SL} aligns to b_{TL}, then there cannot be an alignment where a_{SL} is a dependent of b_{SL}, but b_{TL} is a dependent of a_{TL}. Alignments between dependency structures are grown from the lexical level up, where lexical alignment is done using a bilingual dictionary. The inferred transfer rules are compositional, i.e. they can combine with each other, but they are not typed for constituents, so that they capture the transfer of chunks rather than constituents.

(Lavoie et al., 2002) also start from parsed bilingual (Korean and English) text. The extracted patterns include information on a number of linguistic features such as number or person. At run-time, the induced rules are applied to a parsed Korean sentence, and the resulting tree is passed through an English generation module. While the paradigm employed in this work is also dependency structures, the transfer rules are similar to the ones described in this thesis in that they include "constraints", i.e. context conditions in the form of feature-value pairs. For example, some rules only apply if one of their slots can be filled by a definite noun.

One example of a system that learns transfers of constituent trees is (Kaji et al., 1992) (which actually groups itself into the EBMT framework). Again, the training data is parsed bilingual text, and an algorithm aligns the trees starting from the lexical level with a dictionary, and then extracts transfer rules. Here the variables in the transfer rules are constituents, similar to our approach.

The systems described in this section use parses for both languages and infer transfer rules from aligned trees or dependency structures. Our system is unique in that it learns not only from a much smaller corpus, but also using a parser only for one of the languages.

2.2.3 Translation Templates for Example-Based MT

In the literature on example-based methods, it is often noted that the examples in a bilingual corpus present a possibility for generalization. A large number of researchers have proposed approaches to store the bilingual corpus not simply as a set of examples, but to extract regularities in the corpus. These regularities, or generalizations, can take various forms. Some approaches compare pairs of bilingual sentence pairs to determine what phrases regularly translate into each other, resulting in a phrase- rather than word-based translation lexicon. The SMT community has recently shifted towards learning phrase-to-phrase (rather than word-to-word) trans-

lation rules (Och & Ney, 2004). The generalized EBMT approach differs from the SMT phrase-to-phrase translation in that it stores *templates*. In other words, along with phrase-phrase translation examples, it stores what must come before and after the translation example, e.g. X [$phrase_{SL}$] Y → Y [$phrase_{TL}$] X, where X and Y can be filled by certain other translation examples. For example, (Guvenir & Tunç, 1996) generalize examples by aligning similar and non-similar parts of sentence pairs. The a priori assumption in their work is that similar parts in a sentence translate into similar parts in the other language, and non-similar parts translate into non-similar parts.

A large number of projects extracts generalized patterns using a dependency structure. This has the advantage that the patterns will naturally be lexicalized, which is in line with the spirit of EBMT. Transfer-based systems often put more emphasis on constituent transfer, whereas EBMT systems emphasize more the transfer of specific words. One such system is described in (Alshawi et al., 1998; Alshawi et al., 2000), which learns head transducers from bilingual text. The sentence pairs are word-aligned automatically, and then heads are found by means of heuristics such as word frequency or word length. Finally, the transducers are extracted from the word-aligned dependency structures. The transducers are head lexicalized, meaning that the head word is translated, and the transducer specifies how (e.g., in what order) the dependents are realized in the TL. The advantage of this approach is that it requires no outside resources such as a parser or a morphology module. On the other hand, our approach enables us to generalize further from the training data and to capture more syntactic regularities between languages.

Similar to the 'transfer'-based approaches above, some EBMT systems infer general rules from parsed and aligned bilingual text. For example, (Watanabe et al., 2000) grow rules between source and target language sentences using parses, starting from word alignments. The phrase alignments are restricted in order to avoid overgeneralization.

The rules inferred in these approaches are similar to our approach in some ways. To varying degrees, they include ordering information between constituents (including reordering between languages), generalization to constituents such as NP, PP, etc., typed compositionality where slots in rules can only be filled by other templates of a specific type such as NP. Some include contextual information in the form of linguistic features. This is closely related to the unification constraints learned in our system. It should however be mentioned that in general EBMT learns from large datasets. The rules described in this thesis were learned from a very small dataset, which

sets our work apart from the work described in this section.

2.2.4 Syntax for Statistical MT

Recent years have seen a flurry of activity in incorporating syntactic information into statistical systems. A front-runner in this area was (Wu, 1997), proposing Inversion Transduction Grammars. ITGs represent an exception to the noisy-channel model usually used for SMT. In ITGs, bilingual data is parsed in both languages, and then reorderings of subtrees are learned from the matched parses. Similar projects use parse information for both languages, and learn tree transformations from the training data. (Alshawi et al., 1998), (Alshawi et al., 2000) describe a dependency-based head transducer approach. Their system learns rules that translate a head-word into its corresponding translation, but also translate the dependent words, thus allowing for recursive reorderings.

(Yamada & Knight, 2001), (Yamada & Knight, 2002) focus on string-to-tree transformations. In their approach, syntactic variation is built directly into a statistical translation model. During translation, the system allows with certain (trained) probabilities for translations, insertions, and reorderings. All of these operations are performed on a parse tree.

Similarly, both (Charniak et al., 2003) and (Zhang & Gildea, 2004) learn tree-to-string transformations that are again incorporated directly into the translation model.

(Xia & McCord, 2004) take a similar approach to ours: they learn transfers ('rewrite patterns') using the parses. Their formalism is similar to ours in that the rules are compositional, include word-level, POS-level, and constituent-level elements, and specify alignments. However, the rewrite rules do not include feature constraints. At run-time, the rules are used to modify the incoming SL sentence, so that its ordering will match the TL sentence (similar to the approach described in (Dorr et al., 2002)).

Most recently, (Chiang, 2005) presents an approach that allows for the learning of hierarchical phrase based rules that are very similar in spirit to the rules learned in our approach. Two main differences between Chiang's and our approach are 1) in our system, the phrase types are predefined and learning is only done for those predefined types, and 2) Chiang's learned rules are incorporated directly into the statistical MT system.

Not all efforts to incorporate syntactic information into statistical MT systems have yielded significant results (Och et al., 2003). This indicates that statistical MT systems, although they do not explicitly model syntax, do implicitly prefer grammatical transfer and production, and are very hard

to improve upon. In our work, we observe that the statistical decoder is extremely strong in deciding between partial translations. This is consistent with the observation made in (Och et al., 2003). We remain however convinced that syntax in conjunction with SMT systems can produce superior output. This is confirmed by our results, where we show an improvement in translation quality when grammar rules are used by the system (cf. chapter 9).

A major difference between the approaches described in this section and this thesis is that in our work, we learn transfer rules from extremely small corpora. SMT generally relies on the existence of large bilingual corpora. We show that transfer rules can be learned from much smaller corpora if those corpora are of carefully designed composition.

2.2.5 Other Types of Learned Transfers

As was said above, all of the approaches described here contribute to the blurring of the distinction between traditional approaches to MT. The above approaches could still be loosely classified into transfer, SMT, or EBMT. A few further systems are worth mentioning that do not fall obviously into any of the above classes. For example, (Hermjakob & Mooney, 1997) learn generalized translation rules from shallow parses. In their work, a user trains the parsing system by providing or correcting partial parses, depending on how much the system has already learned. The learned parsing system is used directly to model translation rules.

2.3 Learning from Elicited Data and Small Corpora

In previous sections, we have described systems that learn transfers (or templates, or rules) from bilingual text, generally by taking as input bilingual parsed corpora. In this section, we describe systems that do not rely on the existence of large bilingual corpora, and describe how our work differs from these approaches.

(Nirenburg, 1998; Sherematyeva & Nirenburg, 2000) describe a system, Project Boas, that elicits information from a non-expert user about their native language. For example, with the user's help, it builds an inventory of morphological features that are marked in the language. Furthermore, it elicits subcategorization information. This information is then used to automatically induce transfer rules for MT. The syntax of a language is inferred

by presenting the user with sample constructions in English and asking for a translation, so as to acquire similar structures in the new language. This is similar to our structural elicitation corpus, which is described in chapter 4.

Another system that falls into this category is (Jones & Havrilla, 1998). The paper describes an approach for semi-automatically learning rules for transfer between languages. Similarly to (Nirenburg, 1998; Sherematyeva & Nirenburg, 2000), a user (here, an expert user) provides the system with enough information to infer rules. In this case, the user is presented with a bilingual sentence pair. They are not expected to provide a full parse, but are expected to tag the sentence with linguistic features such as number, and to specify how the words are reordered when translating from the SL into the TL. The approach is called 'Twisted Pair Grammar', because the grammar formalism allows only for reorderings for the children of binary trees, thus restricting the search space.

Although the work described in this thesis uses a carefully constructed, but small corpus, it does not rely on a user to provide linguistic information. In the case of Project Boas, the user provides an inventory of features for their language, whereas in the Twisted Pair Grammar formalism, the user tags training sentences with features. Our work is aimed rather at using a corpus in conjunction with existing resources, i.e. a parser and morphology modules, so as to 1) further automate the inference process, and 2) put less burden on a bilingual user. In the AVENUE project, a bilingual user merely translates and word-aligns the corpus.

Chapter 3

Setting and Run-Time System

In this section, we will describe in detail the Avenue project within which this research was performed. This includes an in-depth discussion of the run-time system complete with a full translation example trace. It should be noted very explicitly that the design and implementation of the run-time system is *not* part of this thesis. It must however be described in order to explain how the learned rules (this thesis) are used to produce translations. For readers who are mostly interested in the rule learning approaches and only wish to get a rough idea of the run-time system, we suggest reading the following two sections (section 3.1, which describes the larger project and the run-time system in brief, and section 3.2, which discusses the rule formalism), and then skipping forward to chapter 4.

3.1 Complete Avenue Project Overview

The rule learner developed for this thesis is part of a larger project, the Avenue project. A system diagram of Avenue can be seen in Figure 3.1. The goal of Avenue is to enable rapid development of MT for languages with few resources. To this end, we elicit data from bilingual speakers using a specifically designed corpus that covers a wide variety of linguistic phenomena, but is kept to minimum size because it has to be translated by an informant. The elicited data, as well as any monolingual data that may be available, can be used to infer morphological information. Within this context, (Monson et al., 2004) have developed a novel technique to detect not only roots and inflected forms, but also inflection classes. Note that

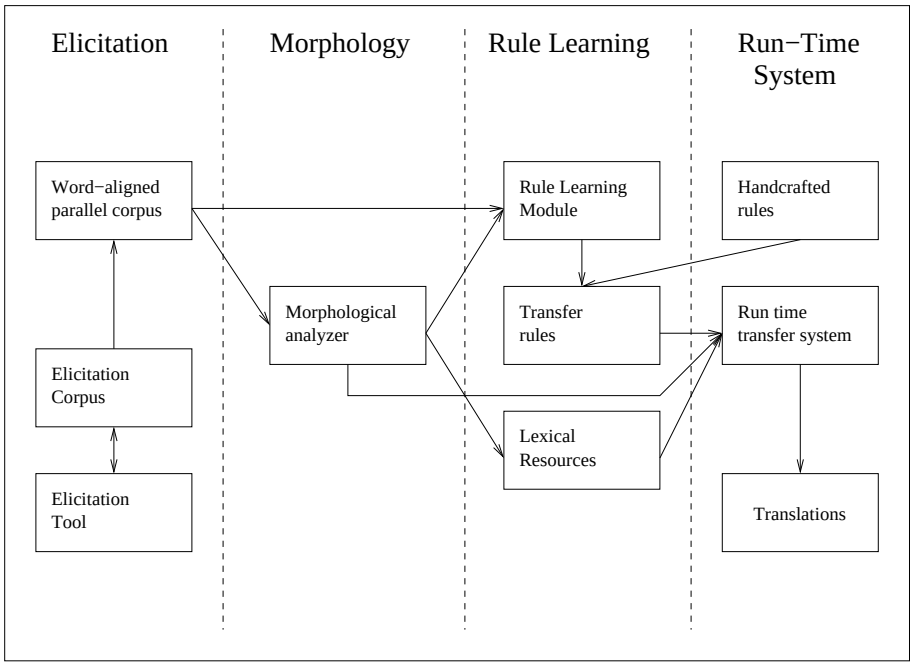

Figure 3.1: AVENUE system diagram.

in the specific language pairs described in this thesis, we used hand-built morphology systems that were available for Hebrew and Hindi, rather than using the system by (Monson et al., 2004). We will discuss this issue in greater detail in section 3.5.2.

The elicited data is however not only used for morphology induction. One other possible application is the detection of linguistic features in a language (Probst et al., 2001), a module that automatically analyzes the elicited data to draw inferences about what features are marked in a language, what parts of speech they are marked on, etc.

The use of elicited data that is most relevant to this thesis is to use the data as training example to the rule learning system (this thesis). The learned rules can either be used as is, or they can go through an iterative refinement step where users that not grammar writing experts can give feedback to the system by means of correcting translations, and the system automatically refines the learned rules based on the feedback from the user (Font-Llitjós, 2004). This is a step that can optionally be added to the rule learning (training time) phase, and is not addressed in this document.

The last step that is necessary for building a functioning system is the creation of a translation lexicon. Such lexicons are often available in some format, and can be typed up if not available in electronic form. If no translation lexicon is available, one can be built from the elicited data (of course, an existing lexicon can also be enhanced using the elicited data). The translation lexicons that were used in the specific systems described in this thesis are discussed in greater detail in section 3.5.3.

The design of AVENUE is very flexible. The learned and possibly refined grammar can always be enhanced by a human expert should one become available. Also, the learned grammar can always be expanded if more data is translated, and can be refined if more feedback is given. The translation lexicon can similarly be expanded on an ongoing basis. This flexible design allows for iterative development of the MT system so that an initial system can be built very quickly and can then be improved over time.

To state clearly the relationship between this thesis and the larger AVENUE project, it should be noted once again that this thesis is *one* piece of AVENUE. More specifically, the AVENUE project ultimately aims at 1) developing resources, such as bilingual corpora and morphology modules, for minor languages, 2) learning and refining transfer rules for language pairs that involve minor languages, and 3) using a combination of manually written and automatically learned and refined transfer rules for Machine Translation in a hybrid transfer-statistical MT system. This thesis is one step on the road to this goal: it develops the underlying technology to *learn transfer rules*. It does not focus on how the resources (corpora, morphology modules) are developed, but rather on how they can be used to learn a transfer grammar. It further does not claim that the learned rules are flawless. Rather, the rules are learned as *one* step towards a full-fledged transfer grammar that can be obtained by 1) refining the rules learned as described in this document, and 2) supplementing them with manually written rules.

The diagram in Figure 3.2 depicts graphically the training time (rule learning) and the run-time (rule application) of the rule learning approach described in this thesis. It shows the different steps that take place during training time, i.e. the learning of transfer rules, and run time, i.e. using the learned rules to produce translations. More specifically, at training time, several resources such as a TL parser and morphology modules are used to learn a grammar (set of transfer rules) from bilingual data. At run time, the learned rules are used by the transfer engine to produce a lattice. Finally, a statistical decoder picks the best partial translations from the lattice to form a translation.

In the following sections, we will describe the transfer rule formalism and

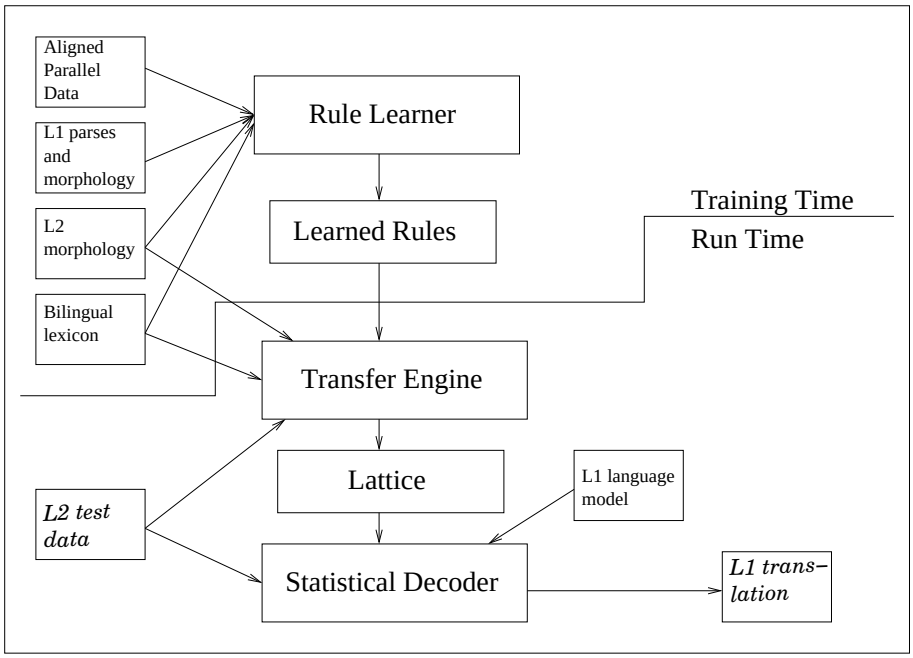

Figure 3.2: Training and run-time of the system. The topic of this thesis is the rule learner.

each of the modules in the larger system.

3.2 Transfer Rule Formalism

The goal of this thesis is to infer structural transfer rules that are enriched with unification constraints. Before delving into the actual learning algorithms, we must explain the transfer rule formalism: what components the rules consists of, what the restrictions on the format of these components are, etc. We will then describe the transfer engine that actually uses the rules to generate a list of partial translations. The decoder then selects partial translations from this list to form a full translation.

The transfer rule formalism is similar to ones that have been discussed in the literature, e.g., (Aho & Ullman, 1969). The transfer rules are designed to be comprehensive and self-contained translation entities: they contain all necessary information to perform analysis, transfer, and generation at run-time, given that an input chunk matches the rule. This is not to say that transfer rules can only be used in isolation: the formalism allows rules to combine with each other, as in a context-free grammar. Structural learning (cf. chapter 7) is in fact aimed at creating rules at different levels of abstraction, so that they can combine in order to apply to a wide variety of contexts.

The comprehensive nature of the transfer rules lends itself especially to learning from bilingual text: each learned rule is always associated with the training example(s) it was derived from. This close tie between the inferred rule and the original training example allows the system to modify, refine, or even eliminate the learned rule during the learning process, always referring back to the bilingual sentence pair that was used to produce it.

The transfer rules are in human-readable format. This is important, as one of the goals of this work was to produce output that a human expert could read, evaluate, refine, and expand on.

An example of a Hebrew→English rule looks as below. We present with it a parallel sentence pair that the rule could translate.

```
Parallel sentence pair:
Hebrew: H   &NIN     H   RB         B       BXIRWT
Gloss:  the interest the widespread in+the  election
English: the widespread interest in the election

Transfer rule:
NP::NP ["H" N "H" ADJ PP] -> ["THE" ADJ N PP]
```

```
(
(X2::Y3)
(X4::Y2)
(X5::Y4)
((Y3 NUM) = (X2 NUM))
)
```

In the rules, the language translated from is the source language (SL) and the language translated into is the target language (TL). In the following discussion, the left-hand side, or the "X"-side, of a rule refers to the SL and the right-hand side, "Y", refers to the TL. In general, we will refer to the left-hand side pieces of information as SL information, and to the right-hand side as TL information. For instance, the NP to the left of the double semi-colon refers to the source language, as does the ["H" N "H" ADJ PP][1], where as the NP to the right of the double semi-colon refers to the target language. In the transfer rule above, the source language is Hebrew, and the target language is English.

In addition to the terms SL/TL and X-side/Y-side, we will use the terms "major" and the "minor" language in our discussion. This will be useful when discussing the different resources that are used for learning, e.g., the English parser. We refer to the major language, English, as TL, and to the minor language, Hebrew and Hindi, as SL, because the run-time system translated from the minor language into the major language. To summarize, the following terms are used as appropriate in context, but refer to the same entities throughout the thesis: **X-side, SL, and minor language**. Further, the following terms refer to the same: **Y-side, TL, and major language**.

In general, every piece of information is given for both languages. Rules consist of a *context-free part* and possibly a set of *unification constraints*. The first context-free component of every transfer rule is the source and target language **type information**, "NP::NP" in the sample rule above, where the first "NP" refers to the SL and the second "NP" to the TL. During learning, this type information is used in classifying transfer training examples, so that only rules of the same type are learned together. At run-time, the type information specifies what slots in other rules a given rule can fill. In this example, the type is NP in both languages, implying that a Hebrew noun phrase translates into an English noun phrase. This has the implications that the translation of the Hebrew noun phrase is assumed to

[1]Entities enclosed in quotes are actual lexical items. This is discussed below in greater detail.

be in some sense similar in composition to the original Hebrew. This will be discussed further in chapter 6.

The type information is followed by a **component sequence**, both for the source and the target languages. We call these sequences component sequences, because they can consist of a combination of lexical items, parts-of-speech, and constituents. In the example rule, the component sequences are ["H" N "H" ADJ PP] for the SL and ["THE" ADJ N PP] for the TL. The components in the sequences can be 1) lexical items (such as 'THE'), 2) part-of-speech labels (such as 'N'), or 3) constituent labels (such as 'PP'). In this rule, the SL component sequence contains the lexical item 'H', and the TL component sequence contains two lexical items, 'THE'. Lexical items are always enclosed by quotes, so that the transfer engine can recognize them as words rather than parts of speech or components. Some learned rules contain lexical items, and we will describe in later sections what causes some words to remain lexicalized in a learned rule. Entities that are not enclosed in quotes are parts of speech (i.e. a N can be filled by any noun) or else generalized constituents (e.g., PP is a prepositional phrase).

At run-time, the SL component sequence is used to parse an input sentence, both to build a tree, and to check if the rule applies to the input sentence or a part thereof. The TL component sequence is used to build the corresponding tree for the TL. This is done by transferring the SL tree into the TL using the **alignments**, e.g. (X2::Y3) above. "Xi" refers to the ith constituent on in the SL component sequence (using 1-based counting). (X2::Y3) means that the second component in the component sequence of the SL (N here) aligns to the third component in the component sequence of the TL (also N). "X0" is reserved for the top level on the source language side (NP here), and "Y0" for the target language top level, also NP here.

It should be noted that the rule encode tree-to-tree transfers at different levels. Each individual rule captures a small tree-to-tree transfer. When the rules are combined, the x-side type and component sequences combine to a large tree which is then mapped to an equivalent tree on the y-side, which is again built from smaller trees from individual rules.

The first two learning phases, Seed Generation and Compositionality, fall into the category of structural learning. Their combined goal is to infer all the rule parts that have been described so far: type information for both languages, component sequences, and component alignments. Further, all the transfer rule components described so far are required. **Unification constraints**, on the other hand, are optional. The above rule has one unification constraint, ((Y3 NUM) = (X2 NUM)). This constraint causes the number value of the second SL component (N) to be passed to the third TL

component (also N). Put more generally, constraints are used as specification information: what sentences does a transfer rule apply to (*x-side constraints*, e.g., ((X2 NUM) = S)[2], what types of features are required when generating the target language sentence (*y-side constraints*, e.g., ((Y1 DEF) = +))[3], and what feature values transfer from the source onto the target language (*xy-constraints*, e.g. ((Y3 NUM) = (X2 NUM)). One important detail about lexicalized items in the component sequences is that the transfer engine does not allow them to be constrained with unification constraints. For example, although factually correct, the constraint ((Y1 DEF) = +) (i.e. the first TL index is definite, which is the case for the definite determiner 'THE') is not allowed by the rule formalism.

While it is useful to classify constraints by what language side they pertain to, it is also necessary for the learning algorithm to distinguish between *value constraints* and *agreement constraints*. As the names indicate, value constraints assign a ground value to the feature of a specific component (index). The constraint ((Y1 DEF) = +) is an example of a value constraint. Agreement constraints, on the other hand, enforce that the values of a specific feature on two components have to agree. ((Y3 NUM) = (X2 NUM)) is an example of an agreement constraint, enforcing that the third English component must agree in number with the second Hebrew component.

In a later chapter on learning unification constraints (cf. chapter 8), we provide a much more detailed discussion of different types of constraints. Constraints can be described with several parameters, such as whether they pertain to a specific word (e.g., Y1 in the example above) or to a constituent (e.g., the PPs (X5 and Y4) in the rule above). Section 8.3 provides an in-depth examination of the types of constraints that can exist, which ones are of interest to our system, and which ones could be useful for other systems or language pairs.

The distinction between context-free and unification components is exploited by learning them separately: first, the system learns the context-free backbone of the grammar. It learns what level of abstraction is appropriate for the learned structures, e.g. when to propose a constituent in the component sequence of a rule (such as the constituent PP in the example above), and when such generalization is better avoided. This step is followed by a step that learns unification constraints: it enriches the context-free rules with whatever constraints are appropriate for the rules. For example, constraints can be used for disambiguation, or to produce more grammatical

[2]This feature is not part of the sample rule.
[3]This feature is not part of the sample rule.

output by enforcing agreement between two components. Three of the main chapters of this thesis (cf. chapters 6 and 7 for structural learning and chapter 8 for Constraint Learning) are devoted to these two separate learning steps. Why is it desired to keep the learning phases separate? One obvious advantage is that the algorithms to learn these two types of information can be designed separately, and can be optimized to the task. This simplifies the design of the algorithms. Another advantage is that the rule learner can be run in different settings. In particular, it can be run without learning constraints, or with learning constraints. As will be described below, if constraints are learned, the run time of the system improves greatly. However, rules without constraints generally lead to higher recall in the translation. This can be more important than run-time efficiency depending on the task, so that Constraint Learning may or may not be desirable. Another, more technical reason to keep the learning modules separate cannot be given until the learning algorithms have been discussed in detail. For this reason, we will put off this discussion until chapter 8.

The type information, component sequences, alignments, and constraints constitute the major rule components. For convenience, we repeat the example rule in Figure 3.3. In this figure, the example rule is marked up with those most essential rule parts. These terms are necessary in order to understand the rest of this document.

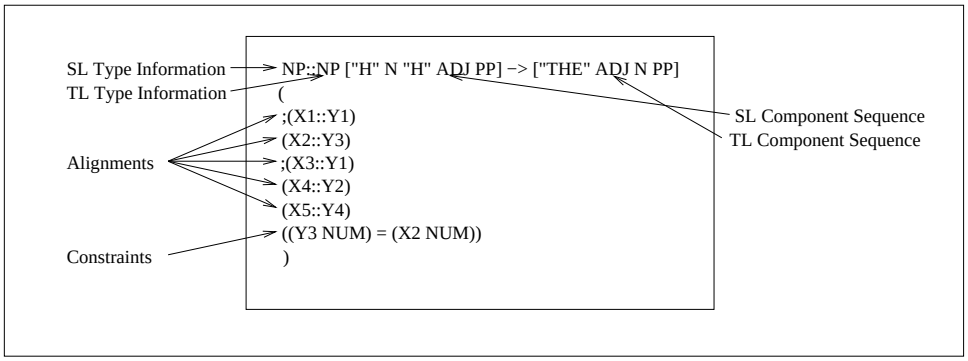

Figure 3.3: Essential rule components.

While the previous rule illustrates the basic format of a rule, several special cases and additions must be discussed. Consider the following rule:

```
;;SL: H N$IM IHIW AIN@LIGN@IWT
;;TL: THE WOMEN WILL BE INTELLIGENT
```

```
;;SL(alt1): H AI$H THIH AIN@LIGN@IT
;;TL(alt1): THE WOMAN WILL BE INTELLIGENT
;;SL(alt2): H AN$IM IHIW AIN@LIGN@IM
;;TL(alt2): THE MEN WILL BE INTELLIGENT
;;SL(alt3): H AI$ IHIH AIN@LIGN@I
;;TL(alt3): THE MAN WILL BE INTELLIGENT
S::S [NP "HIH" ADJP] -> [NP "WILL" "BE" ADJP]
(
(X1::Y1)
;(X2::Y2)
;(X2::Y3)
(X3::Y4)
(X0 = X2)
((Y1 GEN) = (X1 GEN))
((Y1 NUM) = (X1 NUM))
((Y1 PER) = (X1 PER))
(Y0 = Y2)
)
```

One difference between this and the previous rule is that the second rule contains several lines that begin with a semicolon. Any line that starts with a semicolon is ignored by the transfer engine. Such lines generally contain additional information about the rule that is useful for the human reader. In this rule, two examples of such information are given. The first are alignments that are prepended by a semicolon, in this rule ;(X2::Y2) and ;(X2::Y3). These lines reflect the word alignments that were provided by the user. In this case 'HIH' aligns to both 'WILL' and 'BE'. Why are those alignments commented out and others are not? The reason is that the transfer engine considers all lexical items in component sequences by definition unaligned. We sometimes choose to include the alignments for readability as a reminder of the alignments that were specified by the user. The rule learner simply automatically comments out those alignments referring to words that remain lexicalized in the component sequences.

Furthermore, the rule in this example is annotated with all the training pairs that produced it, in each case with the SL (Hebrew) and the TL (English) sentences. If a rule was produced from more than one example, as the rule above, the additional training examples are denoted with "alt" for "alternative".

Note that in this rule, the Hebrew sentence is given in a transcribed romanized form, as will be done for all examples in this document. The

romanization facilitates reading as well as automatic processing, and is a lossless mapping. Hebrew and Hindi romanization is described in greater detail in section 3.5.1.

Finally, it should be noted that sometimes we list different information with the rule as appropriate. For example, in the following rule we list not only the training sentences that produced it, but also their parse and alignments. In chapter 4, we will discuss in detail what information is given in the training data. To understand this rule example, it is only important to know that with each bilingual training example, the rule learner is given the syntactic parse of the English sentence (referred to as the 'C-Structure'), as well as word alignments that are specified by the translator that created the bilingual training data.[4]

```
{NP,16}
;;SL: H &NIIN H RXB B H BXIRWT
;;TL: THE WIDESPREAD INTEREST IN THE ELECTION
;;Alignment: ((1,1),(1,3),(2,4),(3,2),(4,5),(5,6),(6,7))
;;C-Structure:(<NP> (DET the-1)(ADJ widespread-2)
    (N interest-3)(<PP> (PREP in-4)
    (<NP> (DET the-5)(N election-6))))
NP::NP ["H" N "H" ADJ PP] -> ["THE" ADJ N PP]
(
;(X1::Y1)
(X2::Y3)
;(X3::Y1)
(X4::Y2)
(X5::Y4)
((Y3 NUM) = (X2 NUM))
)
```

Note that unlike before, we now include the word alignments of the determiner 'H' being aligned to 'THE', although in commented-out form. Again, these additional pieces of information are given only for the reader and developer, and are not used by the transfer engine, as they are commented out by a semicolon at the beginning of the line. In this document, we give such additional information only when it is useful to the discussion at hand.

[4]The reader may notice that the English 'IN'+'THE' is generally translated into only one Hebrew word, 'B'. In our system, the translator explicitly separates the preposition 'B' from the determiner 'H', so that the words can be aligned separately. This has the effect that the Hebrew training data has the additional word 'H'.

Note that the alignments that are given by the user are stored in a slightly different format from the format that is used in the rule. Here, (1,3) means that the first word of the TL (English) sentence aligns to the third word of the SL (Hebrew) sentence. The direction is reversed: in the original alignments the direction is (English,Hebrew), whereas in the rule, the alignments are given in the direction (Hebrew::English). This is simply an artifact of elicitation being done in the opposite direction of run time translation. This issue is not of great importance to this thesis, but it should be kept in mind when examining the examples.

Further, the original alignments provided by the user are given at the *word* level, but the learned rule gives alignments for the actual *components*, which can be constituents. For example, in this rule, the alignments (4,5),(5,6), and (6,7) as given by the user are reflected in the rule as (X5::Y4), because 'IN THE ELECTION' and its Hebrew equivalent were generalized to a PP.

Finally, this rule is annotated by a unique identifier, in this case "NP,16". The rule learner can automatically assign unique identifiers to rules, which can be used to track the behavior and effectiveness of specific rules. The identifiers do not have any impact on the application of the rule, i.e. the rule would apply in the same way if no identifier (or a different identifier) were assigned to it; it is useful only to the developer. In this thesis, we will use unique identifiers when discussing the behavior of the individual rules in section 9.5.

3.3 Transfer Engine

The transfer engine, the first of two big components of the run-time system, is a module that produces a list of partial hypothesis translations of the input sentence, as licensed by the translation lexicon and the rules in the grammar. It was developed by Erik Peterson (Peterson, 2002). The transfer engine takes as its input a translation lexicon, a set of rules of the formalism described above, morphological information for one or both languages if available, and source language input text, and it creates target language output. The translation lexicon uses a formalism that is very similar to the transfer rule formalism (however, the rules are generally simpler). Two examples of transfer lexicon entries can be seen below:

```
{DET,0}
DET::DET |: ["H"] -> ["THE"]
(
```

```
(X1::Y1)
)

{N,0}
N::N |: ["BIT XWLIM"] -> ["HOSPITAL"]
(
(X1::Y1)
)

{N,1}
N::N |: ["SPR"] -> ["books"]
(
(X1::Y1)
((Y0 NUM) = P)
((X0 NUM) = P)
((Y0 lex) = "BOOK")
)
```

The transfer engine can distinguish lexical entries from grammar rules by the additional symbol '`|:`' in the lexical entries. Like grammar rules, the lexical items can be annotated with unique identifiers (e.g., N,0 here).

The above lexical rules demonstrate that lexical entries are annotated with POS information. This is important when they are used in conjunction with grammar rules. As was discussed above, the rules' component sequences often contain parts of speech, which can be filled by any word in the lexicon of this POS. For example, consider again the rule from the previous section:

```
;;SL: H &NIIN H RXB B H BXIRWT
;;TL: THE WIDESPREAD INTEREST IN THE ELECTION
NP::NP ["H" N "H" ADJ PP] -> ["THE" ADJ N PP]
(
;(X1::Y1)
(X2::Y3)
;(X3::Y1)
(X4::Y2)
(X5::Y4)
((Y3 NUM) = (X2 NUM))
)
```

Here, the slot X2::Y3 could be filled by the second lexicon entry above if 1) it is of the correct POS to fill the slot, and 2) the two components X2 and Y3 align in the rule. N,0 above demonstrates that phrasal entries are allowed in the translation lexicon. Like non-phrasal entries, they can serve to fill 'N' components in the grammar rules.

The rule can optionally contain feature information in the form of unification constraints, e.g., '((Y0 NUM) = P)' in N,1 above. When the lexical entries are applied at run-time, these features are unified with any information that SL morphology modules return. For example, N,0 would not apply if the input sentence contained a singular noun. The transfer engine first runs the input words through the morphology module, and then matches the resulting SL word in *root* form, together with their features, against the lexical entries. This points to another feature of the lexical entries: generally (with exceptions of phrases and fixed expressions), the SL word in the lexical entry is given in the root form, not in the inflected form. This has the advantage that 'generic' lexical entries, i.e. entries without feature information, match all inflected forms of the SL word at run-time, provided that the morphology module can correctly reduce the inflected form to the root form. How the constraints and the root form of the SL word are used in the run-time system will be demonstrated in an example in section 3.6. The specific lexicons used in our Hebrew→English and Hindi→English systems are described in section 3.5.3.

The transfer engine uses the rules to analyze the source language sentence, using a standard bottom-up chart parsing algorithm. (Chart parsing is described e.g., in (Allen, 1995).) The parsing stage results in a set of all those parse trees that are licensed by both the lexicon and the grammar rules. During parsing, ambiguities can be resolved by using the feature constraints in the lexical and grammar rules. Feature information on specific input words is obtained from the lexicon and the SL morphology module. The resulting set of possible analyses is then transferred into the target language via an integrated transfer and generation approach. The system builds structure trees for the target language side, again using the grammar and lexicon to produce all possibilities. During generation, constraints (which are initially marked on specific words, thus at the bottom of the generated trees) are passed up the possible trees as specified by the constraints. This is followed by a constraint checking pass which similarly to parsing eliminates hypothesis translations whose feature values do not match the ones specified in the rules. For all generation trees that were not eliminated in this step, the transfer engine produces target language output. The final product is a (generally large) *lattice* that contains *all* possible partial

translations for a given input sentence. The lattice consists of a set of *arcs*, which are partial translations of the sentence, indexed by what portion of the input sentence they span. Individual arcs do not necessarily cover the entire input sentence; in fact, very few do. Rather, the lattice contains all possible partial translations of all input chunks. Sample lattice entries, i.e. arcs, look as follows:

```
(11 13 "FOUR YEARS" 3 "ARB& $ NIM" "@NP2")
(8 13 "AGO FOUR YEARS" 3.3 "L PN $LI ARB& $ NIM" "@PP")
(8 13 "BEFORE FOUR YEARS" 3.2 "L PN $LI ARB& $ NIM" "@PP")
(8 13 "IN FRONT OF FOUR YEARS" 3.1 "L PN $LI ARB& $ NIM" "@PP")
(8 13 "PRIOR TO FOUR YEARS" 3 "L PN $LI ARB& $ NIM" "@PP")
(8 13 "UNTIL FOUR YEARS" 2.9 "L PN $LI ARB& $ NIM" "@PP")
```

The first two indices indicate what span of the input sentence is covered, followed by a partial translation, a score (that is not currently used by the transfer engine), the translated input chunk, and finally the type of the highest-level grammar rule that was used to produce the output.

So far, we have described the behavior of the transfer engine when a translation grammar is given. However, a translation grammar is strictly optional for transfer engine usage. When no translation grammar is given, the transfer engine can build a lattice using only the lexicon. This reduces to a word-by-word translation, or a phrase-by-phrase translation if phrases are given in the lexicon. The advantage of this flexible design is that it allows us to compare the performance of the run-time system without the learned rules and with the learned rules.

One difficulty that we often face in practice are extremely large lattices. In its default setting, the transfer engine produces a full lattice. This means that it enumerates all possible rule applications and their combinations, as well as combinations with lexical items, for all input lengths. For example, for a sentence of m words, the transfer engine will first create a lattice for each of the m words separately.[5] This is done by using all applicable rules (both in the grammar and the lexicon) to give a translation of each word. Then the transfer engine produces all possible partial translations for bigrams, using a sliding window over the input sentence. Again, all possible rules are applied. This process is repeated until all possible partial translations for all possible n-grams are translated. For example, given an input chunk $w_1w_2w_3$, the transfer engine populates the lattice not only with all possible translations for $w_1w_2w_3$, but also with all translations for w_1,

[5]This is a conceptual, not an actual description of the transfer engine algorithm.

all translations for w_2, all translations for w_3, all translations for w_1w_2, and all translations for w_2w_3. When the grammar is big or ambiguous, each of these chunks can result in a large number of possible translations. Sometimes, the lattice becomes so large that it cannot be produced because of hardware limitations. In such cases, we either make the learning settings more restrictive, or else we produce a partial lattice using a search that is similar in spirit to a beam search: we specify a lengthlimit beyond which the transfer engine does not produce translations. For example, if we set the lengthlimit to 2 for the input chunk $w_1w_2w_3$, then the transfer engine does not produce the possible translations for $w_1w_2w_3$, because this input chunk is of length 3. Instead, it will only produce partial translations for all uni- and bigrams within $w_1w_2w_3$. At first sight, this strategy does not seem to obviously limit the lattice size. The key to understanding why this strategy works is that higher-order n-grams suffer from exponential explosion of all combinations of the lower-order n-grams. For example, the partial translations of $w_1w_2w_3$ are combinations (as licensed by the grammar) of the translations of its unigrams and bigrams. For this reason, the lengthlimit has been found to be an effective strategy. One of the goals of chapter 9 is to assess the impact of different maximum arc lengths (i.e. lengthlimits) on the translations.

Table 3.4 summarizes the inputs and output of the transfer engine.

Transfer engine input and output

- **Inputs:**
  - **Required:**
    1. Translation Lexicon
    2. SL input sentence(s)
  - **Optional:**
    1. Transfer rules (grammar)
    2. SL morphology module
    3. TL morphology module
- **Output:** Lattice of partial translations, indexed by the translated input chunk.

Figure 3.4: Input and output of transfer engine.

The transfer engine is used at run-time as the first step towards producing translations, with the statistical decoder being the second step. However, the transfer engine is also used at training time in order to conveniently assess what rules have already been learned. This information is used when producing compositional rules that are intended to combine with each other. A detailed discussion of this usage of the transfer engine must be postponed until chapter 7. For now, the transfer engine should be thought of only in the context of run-time translation.

The design and implementation of the transfer engine is not part of this thesis. It is part of the run-time system within which the learned rules are applied. For more details, please refer to (Peterson, 2002). However, in order to clarify more how the grammar, lexicon, morphology modules, transfer engine, and decoder interact, we will demonstrate a full sample trace of a run-time translation in section 3.6.

3.4 Statistical Decoder

The statistical decoder is the second phase of run-time translation after the transfer engine, and is another area of investigation that is beyond the scope of this thesis. The specific statistical decoder used in the AVENUE system is the decoder of the CMU SMT system, which is described in (Vogel et al., 2003).

In the first phase, the transfer engine uses the learned grammar (if available), lexicon, and morphology modules (if available) to produce a list of partial translations in the form of a lattice. The lattice is then fed into a statistical decoder that uses a TL language model to find the most likely path through the translation lattice. This is done by a Viterbi-style beam search through the lattice, where combinations of arcs are scored according to the TL language model. Optionally, the lattice entries can be scored according to a translation model if available. The decoder allows for reorderings of words that were translated using only the lexicon, but it does not reorder words within chunks that were produced using transfer rules. In this way, the confidence in the chunks produced by grammar rules is raised. This is important, because otherwise the decoder could choose to reorder within the partial translations. This would mean that the reorderings captured in transfer rules and expressed in partial translations would be ignored, which is clearly not a desirable effect. Figure 3.5 summarizes the input and output of the statistical decoder.

Statistical decoder input and output

- **Input:** Lattice of partial translations, indexed by the translated input chunk.
- **Output:** Full translations, one per sentence.

Figure 3.5: Input and output of statistical decoder.

3.5 Language Pair Specific Components

In this chapter, we discuss components or modules that are specific to the Hebrew→English and Hindi→English run-time systems. Although the actual run-time systems are not part of the thesis work, we describe them here in detail so that the user may obtain a better idea of how things work in practice. As was said before, the reader who is merely interested in the rule learning algorithms may wish to skip ahead to chapter 4.

3.5.1 Character Sets and Romanization

The first language-specific topic to be addressed is romanization. As has been seen before, we present all Hebrew and Hindi examples in this thesis in romanized form. Both Hebrew and Hindi use character sets that are different from Latin-1. In order to standardize and simplify things and because some of the components in our systems expect romanized representations, we chose to romanize the Hebrew and Hindi characters.

Romanization in the Hebrew→English system

The Hebrew character set consists of 22 characters, plus 5 characters that are variations of one of the 22 characters and are only used at the end of words. In other words, 5 characters are represented differently when it occurs at the end of a word vs. when it occurs at the beginning or in the middle of a word. In our system, we ignore this difference between the end of a word and the beginning or middle of a word, which leaves us with 22 characters. Table 3.1 lists the mappings of the 22 characters to a Roman character set. For the interested reader, we also list the common pronunciations of each of the characters, both in intuitive description and in the representation in the International Phonetic Alphabet.[6]

[6]Information taken from http://www.omniglot.com/writing/hebrew.htm

The mapping in this table is not the only existing mapping of Hebrew characters to Roman characters; several other such transliteration tables exist. We chose to use this particular mapping, because it is the one that is used by the Hebrew morphology module in our system. The Hebrew morphology module is described in section 3.5.2 below.

One complication with Hebrew script is that vowels are not generally written, with some exceptions. For this reason, Hebrew textual input is often highly ambiguous, because different vowel combinations may indicate differences in POS, tense, number, etc.

Romanization in the Hindi→English system

The Hindi Devanagari character set contains 33 consonant characters, as well as 28 vowels and diacritics. In addition, several characters are used only in loanwords, are combinations of characters, or special characters. Finally, each of the numerals (including 10) has a character representation in Hindi.[7]

The Devanagari character set cannot be simply mapped to a romanized representation, because the characters take different shape depending on the surrounding characters. Quite a number of character representations exist for Hindi. The one that we chose to use is the romanized character representation RomanWX. RomanWX is routinely used in computer processing of Hindi.

Several converters exist between UTF8 (in which Devanagari is represented) and other representation such as RomanWX. One such converter can be found at http://sarovar.org/projects/codeconverters. These tools also provide additional information on the mapping between Devanagari characters and the RomanWX representation.

3.5.2 Morphology Modules

In our Hebrew→English and Hindi→English systems, we use morphology modules that were developed with hand-written rules. It was important for the rule learning module (this thesis) to have access to morphological information in the minor languages. However, as we have emphasized above, it is not necessary to have hand-built morphology modules available for all language pairs. Hand-built systems will be higher in quality, but they may not be available for all language pairs. As the rule learning simply uses the output of morphological analysis as one of its inputs, the morphological

[7]Information drawn from http://www.omniglot.com/writing/hindi.htm.

Letter Name	Roman	Pronunciation	IPA
Alef	A	’	[ʔ,ø]
Bet	B	b/v	[b,v]
Gimel	G	g	[g]
Dalet	D	d	[d]
He	H	h	[h]
Vav	W	w	[v]
Zayin	Z	z	[z]
Het	X	h/ch	[x]
Tet	@	t	[t]
Yod	I	y/i	[j]
Kaf	K	k/kh	[k,x]
Lamed	L	l	[l]
Mem	M	m	[m]
Nun	N	n	[n]
Samekh	S	s	[s]
Ayin	&	’	[ʔ,ø]
Pe	P	p/f	[p,f]
Tsadi	C	tz/ts/z	[ts]
Qof	Q	k	[k]
Resh	R	r	[r]
Shin	$	sh/s	[ʃ,s]
Tav	T	t	[t]

Table 3.1: Mapping of Hebrew characters to Roman representation. We list here also the pronunciation of Hebrew characters in International Phonetic Alphabet (IPA).

information could also stem from a system that was built automatically, i.e. an unsupervised morphology module for the minor language.

Unsupervised or minimally supervised learning of morphology is a growing research field with quite a number of researchers addressing the problem. Generally, the task is defined as automatic assignment of inflected forms to their roots as well as the isolation of morphemes in a language. (Goldsmith, 2001) and (Snover & Brent, 2002) use the context or paradigmatic structure of words to accomplish this. Other approaches, such as (Johnson & Martin, 2003) use a finite state approach to detecting morphemes, whereas (Schone & Jurafsky, 2001) view the problem as an LSA problem. What all these approaches have in common is that they do not consider it part of the task to assign meaning to morphemes. This, however, is a crucial step if a morphology module is to be used as underlying technology for our approach. For example, in many cases it is useful to know that 'S' is a morpheme in English, but for our task and other MT tasks, we also need to know that 's' can either mark a noun for plural or a verb for 3rd person singular present tense. How can this be accomplished? Several approaches have been suggested that move research into this direction. For instance, (Yarowsky & Wicentowski, 2000) propose several approaches that allow the reduction of specific inflected forms (such as past tense) to root forms. This approach requires knowledge of the morphology of a language. In other words, it must be known a priori what features are marked in the language. The advantage of the approach is however that the detected inflection forms are immediately mapped to meaning. (Yarowsky et al., 2001; Yarowsky & Ngai, 2001) as well as (Probst, 2003) use word alignments and projection from one language into another to infer some morphological information and POS tags for another language. This approach has a similar advantage of immediate assignments of morphemes to meaning, but it requires a (large) bilingual corpus, which is not necessarily given for all languages. Within the AVENUE project, (Monson et al., 2004) have developed a novel technique to detect not only roots and inflected forms, but also inflection classes, thus deriving more information about the language's morphology system. While all these approaches provides steps in a direction that would make an automatically inferred morphology module useful to our approach, no single technique as yet is powerful enough, and more research is necessary. For this reason, the work described here relies on the existence of a manually developed morphology module. Currently, such morphology modules are available for a variety of languages (including Hebrew and Hindi, which are of relevance to this thesis), or can be developed for Mapudungun or Quechua, as is being done within AVENUE.

Hebrew morphology in the Hebrew→English system

The Hebrew morphology module is a freely available tool described in (Itai & Segal, 2003) and available at (Segal, 2001). The tool expects input in romanized form (the mapping from Hebrew script to Roman characters was described in the previous section). One of the difficulties in handling Hebrew is that both the definite determiner 'H' and prepositions such as 'B' (in, at), 'M' (from) etc. are generally merged with the noun that they precede. For example, 'the house' would be represented by only one word in Hebrew, 'HBIT'. During training, we circumvent this problem by separating this word into two, where one represents the determiner 'H' and one represents the noun 'BIT'. During run-time, however, the morphological analyzer must separate merged input words. To complicate matters further, the Hebrew 'LILD' ('to the/a boy) is ambiguous: the preposition 'L' (to) merges with the definite determiner if one is present. In other words 'L'(to)+'H'(the)+'ILD'(boy) would be written as 'LILD'. Indefiniteness is not marked in Hebrew, so that the phrase 'to a boy' would be represented by 'L'(to)+'ILD'(boy), which would also merge to 'LILD'. The task of the morphological analyzer is then to propose all ambiguities. When faced with a possibly ambiguous tokenization, the morphological analyzer proposes all analyses for all possible tokenizations. For example, 'LILD' is analyzed by the morphology module to have the possible tokenizations 'LILD', 'L'+'H'+'ILD', and 'L'+'ILD'.

The analyzer returns a number of feature values for each word. First, it returns the set of possible root forms for this word. For example, if the input word is an inflected word, the morphological analyzer returns the citation form of this word in Hebrew. This is important (among other things) because for most entries in the Hebrew→English dictionary, the Hebrew word is in the root form. It is thus necessary to analyze each Hebrew input word with the morphology module, so that the root form can then be matched against the entries in the dictionary, and can then be used in grammar rules.

As was said above, Hebrew does not generally write vowels, so that a given word can often be analyzed as more than one part of speech. All possible analyzes are returned by the morphology module.

In addition to POS information, the analyzer also returns a number of feature-value pairs as appropriate. These features include gender, number, status (for noun compounds), person, and tense information.

When a word cannot be analyzed, the morphology returns an 'analysis' with a POS 'LEX', which is used for all words for which the POS is unknown. No features are returned for a 'LEX' analysis. As was described above, each

input word is analyzed in all possible tokenizations, so that a 'LEX' is often given for a non-tokenized word.

Sample output from the morphology module for the PP 'LILD' can be seen in Figure 3.6. Here, the 'SPANSTART' and 'SPANEND' indices indicate what part of the tokenized word is spanned by a given analysis. For the word 'LILD' the maximum tokenization 'L'+'H'+'ILD' determines the number of indices. This means that if an analysis has SPANSTART 0 and SPANEND 1, then it spans the word 'L'. If it has SPANSTART 2 and SPANEND 3, then it spans 'ILD'. This is illustrated in the following table:

Index 0-1	**Index 2-3**	**Index 3-4**
L	H	ILD

The morphological analyzer analyzed the word 'LILD' in several ways: it recognized that the preposition 'L' can be split off, and that the definite determiner 'H' can be inserted, resulting in an analysis of a definite prepositional phrase 'L'+'H'+'ILD' (to the boy). However, the analyzer also recognized the analysis where the definite determiner 'H' is not present, resulting in an analysis of 'L'+'ILD' (to a boy). Further, the noun 'ILD' could be in either construct or absolute status. The status feature indicates what part of a noun *compound* a word can be (because nouns change their form in noun compounds under certain circumstances). The final analysis of 'LILD' is one that spans the entire word. No analysis was found for this word, so that the analyzer simply returns a 'LEX' as the POS and no features. It should finally be noted that the scores that are returned by the morphology module are not currently used in our system.

The transfer engine handles the ambiguity in morphology by passing on all possible morphological analyses. This means that all analyses are used in rule applications and lexical transfer.

Hindi morphology in the Hindi→English system

The morphology system that is used in our Hindi→English translation system is, like in the Hebrew case, readily available software that was not developed within the Avenue project. The particular Hindi morphology module that is used in our system was developed at IIIT in India and can be obtained from http://www.iiit.net/ltrc/morph/index.html.

Like the Hebrew morphology module, the Hindi morphological analyzer takes as input fully inflected Hindi words. It returns first the root form

```
Doing morphological analysis on LILD
Y0: ((SPANSTART 0)
     (SPANEND 1)
     (SCORE 1)
     (LEX L)
     (POS PREP))
Y1: ((SPANSTART 1)
     (SPANEND 2)
     (SCORE 1)
     (LEX H)
     (POS DET))
Y2: ((SPANSTART 2)
     (SPANEND 3)
     (SCORE 363)
     (LEX ILD)
     (POS N)
     (GEN M)
     (NUM S)
     (STATUS ABSOLUTE))
Y3: ((SPANSTART 0)
     (SPANEND 2)
     (SCORE 1)
     (LEX L)
     (POS PREP))
Y4: ((SPANSTART 2)
     (SPANEND 3)
     (SCORE 337)
     (LEX ILD)
     (POS N)
     (GEN M)
     (NUM S)
     (STATUS ABSOLUTE))
Y5: ((SPANSTART 2)
     (SPANEND 3)
     (SCORE 298)
     (LEX ILD)
     (POS N)
     (GEN M)
     (NUM S)
     (STATUS CONSTRUCT))
Y6: ((SPANSTART 0)
     (SPANEND 4)
     (SCORE 1)
     (LEX LILD)
     (POS LEX))
```

Figure 3.6: Sample Hebrew morphology output.

of the word, which is given in UTF8 and in the romanized (RomanWX) versions. The root form can then be matched against entries in the bilingual dictionary, where most Hindi words are represented in their root form.

The morphological analyzer further returns a number of features for each word, including POS, number, tense, gender, person, and verb form information. Similarly to Hebrew, several different analyses are possible for many words. The transfer engine handles this ambiguity by passing on all possible analyses, as in the Hebrew system.

A sample analysis of the Hindi word is given in Figure 3.7. This example was also given in a previous paper about the Hindi→English system (Lavie et al., 2003). The analyzed word (RomanWX transliteration 'raha', meaning 'continue/stay/keep'). It can be seen in this figure that two analyses are proposed by the morphological analyzer, one where the input word is analyzed as a verb, and one where it is analyzed as a form of the Hindi light verb 'RAHA'.

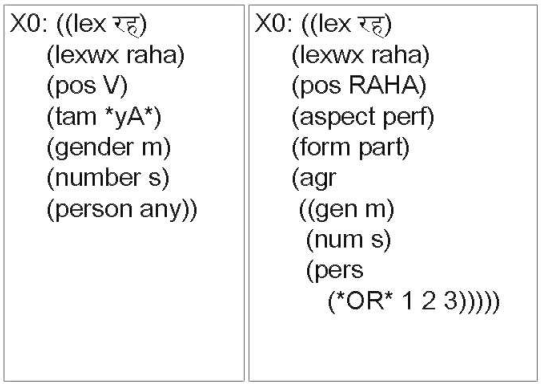

Figure 3.7: Example of Hindi morphology output.

3.5.3 Translation Dictionaries

The translation dictionaries accomplish lexical transfer, as well as feature value passing at the lexical level (in addition to lexical level feature passing in the grammar rules). In both the Hebrew and the Hindi cases, the bilingual lexicons are made up of entries that contain not only the SL (Hebrew or Hindi) word and its English equivalent, but also the POS of the two corresponding words, and whatever feature values apply.

The SL words are given in their root form in the dictionary. As was

mentioned above, all input is first passed through the appropriate morphology module. Then the returned root forms are matched against all lexical entries in order to accomplish lexical transfer. One exception to this rule is the case of phrasal entries, i.e. transfers of sequences of words to sequences of words. The transfer of phrasal entries is only done as a whole, i.e. the entire phrase must match.

Since neither system uses an English morphology generation module, the lexicons are enhanced to contain different forms of English words, together with appropriate features. For example, for the Hebrew word 'BIT' ('house'), there are two entries in the lexicon, one for singular and one for plural. The SL word is, in both cases, given in the root form ('BIT'), but one entry translates into the singular 'house', the other into the plural 'houses'. The transfer is constrained by features, so that only the correct translation is produced. The relevant lexical rules are as follows:

```
N::N |: ["BIT"] -> ["HOUSE"]
(
(X1::Y1)
((X0 NUM) = S)
((Y0 NUM) = S)
((Y0 lex) = "HOUSE")
)

N::N |: ["BIT"] -> ["houses"]
(
(X1::Y1)
((Y0 NUM) = P)
((X0 NUM) = P)
((Y0 lex) = "HOUSE")
)
```

The English word forms that were introduced in the lexicon are plurals for nouns and gerund, 3rd.sg.present, past tense, and past participle for verbs. The English verb forms were generated automatically, where the English 'generation' module (a small in-house module) takes care to observe spelling rules and irregular forms as much as possible. A few misspellings and incorrect forms still exist in the lexicons, but most entries are correct English forms.

The same English 'generation' principle is applied in the Hindi→English dictionary in order to accomplish the generation of different English forms.

The translation lexicon plays a very important role in the application of the learned rules, and thus in the performance of the run-time system. The learned rules rely on the existence of a lexicon that accomplishes not only lexical transfer, but also contains the parts of speech of the relevant words. If an input word is found in the lexicon, it can be translated, but only if it is found with part-of-speech information can it be used in rules. Note that the rules often generalize to applying to a whole class of words of a given part of speech. For example, the sample rule in Figure 3.3 captures the transfer of a N into an N (by the alignment (X2::Y3)). This means that this slot can only be filled by words that are marked specifically as nouns in the dictionary. They must be marked as such in the dictionary.

Hebrew→English dictionary

The Hebrew→English bilingual dictionary contains a total of 62038 entries that were in part derived from elicited data. The largest part of the dictionary was however built based on the Dahan dictionary (Dahan, 1997). The Dahan dictionary contains a Hebrew→English part and an English→Hebrew part. We used both parts (in reversed direction in the case of English→Hebrew entries). The lexicon was also enhanced with some entries that were found missing. Finally, some entries were removed, because their application produced obscure translations that merely increased lexical ambiguity without any added benefit.

Hindi→English dictionary

The Hindi→English dictionary was based to a large extent on a dictionary provided by the Linguistic Data Consortium. As this dictionary is highly ambiguous and contains a number of obscure translations, we eliminated some possible English translations (as in the Hebrew case) in order to eliminate lexical ambiguity and in order to rule out some obscure translations. In addition, we added several types of manually produced rules. First, we extracted lexical transfer rules from elicited data. As the bilingual informants not only translate data, but also specify word alignments, lexical rules can easily be extracted from the hand-translated and -aligned data. In addition, we added several hand-written rules for specific fixed expressions. For a full breakdown of lexical rules, please refer to (Lavie et al., 2003). The current full dictionary contains 57960 entries.

3.6 Run-time Example

In this section, we provide a simple, but complete example trace of how translation is accomplished in the run-time system. We give the run-time example only for Hebrew→English translation; the Hindi→English run-time system behaves sufficiently similarly (for the purposes of this discussion) to make a second trace unnecessary. Also, the trace is slightly simplified and conceptualized, i.e. it does not completely reflect the actual optimized order of operations in the algorithm, but rather explains what the algorithms accomplish. For more details, please refer to (Peterson, 2002). We trace here only the transfer engine phase, as the statistical decoder is a standard statistical decoder described in various places, e.g., in (Vogel et al., 2003).

It should be emphasized that the example in this section is a very simple one. It is merely meant to illustrate the behavior of the system. The main reason to not present a more complex example here is that the complexity increases very quickly, which would lead to a very lengthy discussion without any added benefit or insight. For this reason, we restricted ourselves to a simple example, and trust that it will give the reader an idea of how the system behaves under 'friendly' conditions, and impress upon the reader again that in practice, the system handles much larger grammars and lexicons, as well as much more complex sentence constructions. This will become clear in the remainder of this thesis.

In this example, we will trace the translation of the following phrase:

```
H   SPRIM   H   IRWQIM
the book.pl the green.pl.m
'THE GREEN BOOKS'
```

Let us begin by presenting the lexicon that is used in this example. It consists of only 5 entries:

```
{DET,0}
DET::DET |: ["H"] -> ["THE"]
(
(X1::Y1)
)

{ADJ,0}
ADJ::ADJ |: ["IRWQ"] -> ["GREEN"]
(
(X1::Y1)
```

```
)

{N,0}
N::N |: ["SPR"] -> ["BOOKS"]
(
(X1::Y1)
((Y0 NUM) = P)
((X0 NUM) = P)
((Y0 lex) = "BOOK")
)

{N,1}
N::N |: ["SPR"] -> ["BOOK"]
(
(X1::Y1)
((X0 NUM) = S)
((Y0 NUM) = S)
((Y0 lex) = "BOOK")
)

{N,2}
N::N |: ["SPR"] -> ["BORDER"]
(
(X1::Y1)
((X0 NUM) = S)
((Y0 NUM) = S)
((Y0 lex) = "BORDER")
)
```

We selected these dictionary entries specifically only for this trace. As was said above, our actual run-time system has many thousand (not just five) lexicon entries. The toy dictionary in this example contains three entries for the root of 'SPR' (book). One is for a singular translation, one for a plural translation, and one is for a lexically ambiguous translation into 'border'. This alludes to the problem that the run-time system must often choose between many ambiguous translations.

The grammar for this example is equally simple and consists of only two rules:

```
{NP,0}
```

```
;;SL:H $NH H RA$WNH
;;TL:THE FIRST YEAR
NP::NP ["H" N "H" ADJP] -> ["THE" ADJP N]
(
;--;(X1::Y1)
(X2::Y3)
;--;(X3::Y1)
(X4::Y2)
(X0 = X2)
((Y3 NUM) = (X2 NUM))
((Y3 PER) = (X2 PER))
(Y0 = Y3)
)

{ADJP,0}
;;SL:I$N
;;TL:OLD
ADJP::ADJP [ADJ] -> [ADJ]
(
(X1::Y1)
)
```

Note that both of these rules are actual automatically learned rules that were produced by the system described in this thesis. However, they represent only a very small fraction of the learned rules. The two rules NP,0 and ADJP,0 accomplish the transfer of definite Hebrew NPs that are modified by an ADJP. Note that in Hebrew, the definite determiner is marked on the noun as well as on the adjective. The ADJP rule is the most basic ADJP rule that simply states that a single adjective can function as an ADJP.

These grammar rules cover exactly the phrase that we trace in this example. Note again that in real examples, translation becomes more complex, among other things because many grammar rules can apply to a given input, especially when the input is morphologically ambiguous and can be of multiple POSs. Lexical ambiguity further complicates translation, because it forces the transfer engine to try all possible combinations of lexical choices.

3.6.1 Parsing

During the parsing step, the transfer engine first runs all words through morphology. For example, for the word 'SPRIM', the following two analyses

are returned:

```
X0: ((SPANSTART 1)
     (SPANEND 2)
     (SCORE 1000)
     (LEX SPR)
     (POS N)
     (GEN M)
     (NUM P)
     (STATUS ABSOLUTE))
X0: ((SPANSTART 1)
     (SPANEND 2)
     (SCORE 1)
     (LEX SPRIM)
     (POS LEX))
```

The first analysis correctly analyses the word as a plural form of the word 'SPR' ('THE BOOK'). The second is a default analysis without any feature values. Both of the analyses, i.e. both of the values for the 'LEX' field, are then matched against the dictionary entries, and it is checked whether the features unify with the features that were returned from morphology. For example, the plural analysis of 'SPRIM' with its root (LEX) 'SPR' matches three lexicon entries: N,0 ('BOOKS'), N,1 ('BOOK'), and N,2 ('BORDER'). However, only one entry results in a successful unification, namely N,0 ('BOOKS'). The reason is that N,1 and N,2 are both marked for singular, whereas the morphology analysis for this 'SPR' is marked for plural. For the successful unification, the morphological features are retained for further analysis. In other words, the features are used for any rule application in which this analysis is used. The unified entry is also marked with the POS that is given in the lexicon entry if it is not the same as the POS returned by morphology. The second morphological analysis with LEX 'SPRIM' does not match any lexicon entries.

The same lookup/matching scheme is used on all input words: they are passed through morphology, then matched against the lexicon entries. The input words can also be matched against the lexical entries in their surface form, i.e. before passing them through morphology first to obtain the root form.

Parsing, of course, consists of more than simple lexical analysis. During this step, the transfer engine also tries to 'put together' the lexical analyses by means of the transfer rules in the grammar.

This is done by using the analyses (unifications of morphology and lexical entries, or only lexical entries) with their POSs and their feature-value pairs in the rules.

In this case, the rule ADJP,0 is used to parse the single input adjective 'IRQIM' ('GREEN'.m.pl). In other words, the adjective is analyzed as an ADJP, and the morphological features from the adjectives are passed to the ADJP. The non-terminal node is used in the rule NP,0 to fill the slot X4. When rules are used in combination, it must again be the case that their features unify. In the case of this rule combination, the features values that must unify are those marked on the X0 node on the ADJP rule, and the ones marked on the X4 node for the NP rule.

As was said before, the transfer engine produces all translations (as licensed by the lexicon and grammar) of all input chunks. In order to explain the behavior of the transfer engine, we will focus in this discussion on only one partial translation that is produced. In section 3.6.3 we give the full list of partial translations that were produced for this phrase. For discussion, we focus on the following parse tree:

```
(NP ((‘‘H’’) (N) (‘‘H’’) (ADJP (ADJ))))
```

3.6.2 Transfer and Generation

Transfer and generation are done in an *integrated* step in the transfer engine. However, it is easier to conceptualize the behavior of the transfer engine by discussing transfer and generation separately.

Syntactic Transfer

The first phase of transfer and generation creates a syntactic tree for the TL from each parse tree. This is done using the component alignments that are given in the rules that were used to produce the parse tree. During syntactic transfer, the transfer engine builds a TL tree for all components in the parse tree, except the lexical items. Lexical transfer is accomplished in the subsequent phase.

Syntactic transfer begins from the root node of the parse tree. The transfer engine creates a corresponding TL node. The features that are associated with this node are obtained using the xy- and y-side constraints in the rules that were used to obtain the parse tree.

The process is repeated for the root's children, and recursively after that. However, when producing corresponding nodes for each interior node in the parse tree, the transfer engine uses the alignments in order to put the TL

components in the correct order. This implicitly accomplishes component reordering wherever necessary. Lexicalized components are also handled during this phase: lexicalized items in the SL component sequence are not transferred because they are unaligned. Lexicalized items in the TL component sequence are inserted in the correct place. Again, the transfer engine uses the alignments and component sequences of the rules that were applied to form the parse tree. In our example, we can picture the SL parse tree and the corresponding TL tree as follows:

```
(NP::NP (("H"::"")
         (""::"THE")
         (N::N)
         ("H"::"")
         (ADJP::ADJP (ADJ::ADJ))))
```

This bilingual tree demonstrates how literals are deleted and inserted during syntactic transfer, and how the components are transferred recursively. Each TL node is now associated with a set of features that was passed to it from the corresponding SL node by means of xy- and y-side constraints. In our case, the TL node marked by N is now marked for the number value plural and person value 3rd, as these features were passed to it by the following xy-constraints from rule NP,0:

```
((Y3 NUM) = (X2 NUM))
((Y3 PER) = (X2 PER))
```

The transfer engine does not actually represent the trees as bilingual trees, but rather simply produces a TL tree:

```
(NP ("THE")(ADJP (ADJ))(N))
```

The resulting TL tree does not yet have lexical items associated with the pre-terminal (i.e. POS) nodes. In other words, in our example, although the literal 'THE' has been inserted, the nodes marked as N and ADJ are not yet filled with lexical items. This happens during lexical transfer and generation in the following phase.

Lexical Transfer and Generation

During the final phase of transfer, the TL tree's nodes are filled with lexical items. During parsing, only a subset of the lexical entries that match a SL

word can be applied, because some may result in failed unifications with constraints in the transfer rules. During lexical transfer, a similar principle applies. From those lexical entries that were in fact applied during parsing, the transfer engine now forms a list of possible lexical items for each POS node, in our case for the N and the ADJ nodes. Then it unifies the features associated with the lexical entries with the features associated with the TL tree nodes (here, N and ADJ). This generally results in a smaller list. In our case, only one lexical item remains for each of the nodes, 'GREEN' for the ADJ node and 'BOOKS' for the N node, because the other entries did not unify at parsing time. The transfer engine then proposes all possible combinations of the remaining lexical items. This involves applying all y-side constraints, so that some features are passed up the TL tree from the lexical level. Since higher-level constraints sometimes result in a failed unification (e.g., a singular verb will not unify with a plural noun in subject position, provided the correct constraints are given), some combinations can be eliminated.

After narrowing the list of possible combinations by means of the features and constraints, the transfer engine finally reads the lexical items off the TL tree. The lexical items are already in the proper TL order, and are output as partial translations.

3.6.3 Resulting Lattice

The final resulting lattice for this example is:

```
(
(0 0 "THE" 1 "H" "(DET,0 "THE")")
(1 1 "BOOKS" 1 "SPR" "(N,0 "BOOKS")")
(1 1 "BOOKS" 1 "SPR" "((NP,0 (N,0 "BOOKS") ) )")
(2 2 "THE" 1 "H" "(DET,0 "THE")")
(3 3 "GREEN" 1 "IRWQ" "(ADJ,0 "GREEN")")
(3 3 "GREEN" 1 "IRWQ" "((ADJP,0 (ADJ,0 "GREEN") ) )")
(0 3 "THE GREEN BOOKS" 1 "H SPR H IRWQ" "((NP,0 (LITERAL "THE")
    (ADJP,0 (ADJ,0 "GREEN") ) (N,0 "BOOKS") ) )")
)
```

Each line represents an 'arc', i.e. a partial translation. In this lattice, we give the full trace of each arc. This means that the last field in each arc (i.e. the field enclosed by '"(' and ')"') specifies how the rules were applied together.

In this lattice, we can note several interesting points:

1. There are arcs of different lengths. The transfer engine translates all input chunks of all lengths, unless a maximum length is specified, which we did not do in this example. For this reasons, we have arcs of length 1 and one arc of length 4 (number of indices spanned in Hebrew input).

2. The last arc indicates the rule application of NP,1 and ADJP,1. They were applied in combination, where the actual lexical transfer was handled with rules from the lexicon.

3. There are no arcs that use the lexical rules N,1 and N,2. This stems from the fact that the input word 'SPRIM' was passed through morphology, producing the following analysis:

```
X0: ((SPANSTART 1)
     (SPANEND 2)
     (SCORE 1000)
     (LEX SPR)
     (POS N)
     (GEN M)
     (NUM P)
     (STATUS ABSOLUTE))
```

   In other words, the morphology returned that the word 'SPRIM' is a plural noun. For this reason, the unification with the lexical entries N,1 and N,2 does not succeed, as both of them require that the SL word (X0) is marked for singular rather than plural.

4. The lattice contains duplicates, i.e. arcs that match both in span and in translation. In this case, the duplicates are 'GREEN' and 'BOOKS'. When we do not wish to output the trace and inspect all possible outcomes, then we generally run the transfer engine in a mode that automatically eliminates such duplicates in order to limit the lattice size.

Chapter 4

Training Data

The AVENUE project targets language pairs with an imbalance in electronic resources. In particular, it is not expected that for a given language pair, a large bilingual corpus will be available. For this reason, we are designing a unilingual corpus, generally referred to as the *Elicitation Corpus*. The elicitation corpus is a collection of sentences and phrases in the major language (TL). Most of the designed elicitation corpus so far is in English, but part of it has been translated into Spanish, which will in particular allow us to build MT systems between Spanish and minor languages in South America (for example, Mapudungun). A native speaker of the low-density language of interest (SL) then translates the elicitation corpus into their language, and specifies the word alignments. This is done using an *Elicitation Tool*. The interface was developed by Erik Peterson and is described in (Lavie et al., 2003). It presents users with one sentence to translate at a time. The user enters the translation into a specified line. Word alignments can be specified by clicking on an English word and then the corresponding word in the other language. Not only one-one, but in fact zero-one, one-zero, many-one, one-many, and many-many alignments are allowed in the translation tool. Further, the user is able to give as many alternative translations for a sentence as desired. The interface offers a convenient and intuitive way to do this. A snapshot of the interface can be seen in Figure 4.1.

Although the elicitation tool is very flexible in what word alignments can be given by the user, we encourage translators to abide by certain guidelines when giving alternative translations and when giving word alignments. Alternative translations should be given when a real syntactic alternative exists, or when both alternatives are roughly equally common. It is not necessary to give all alternatives when they only differ in lexical items that

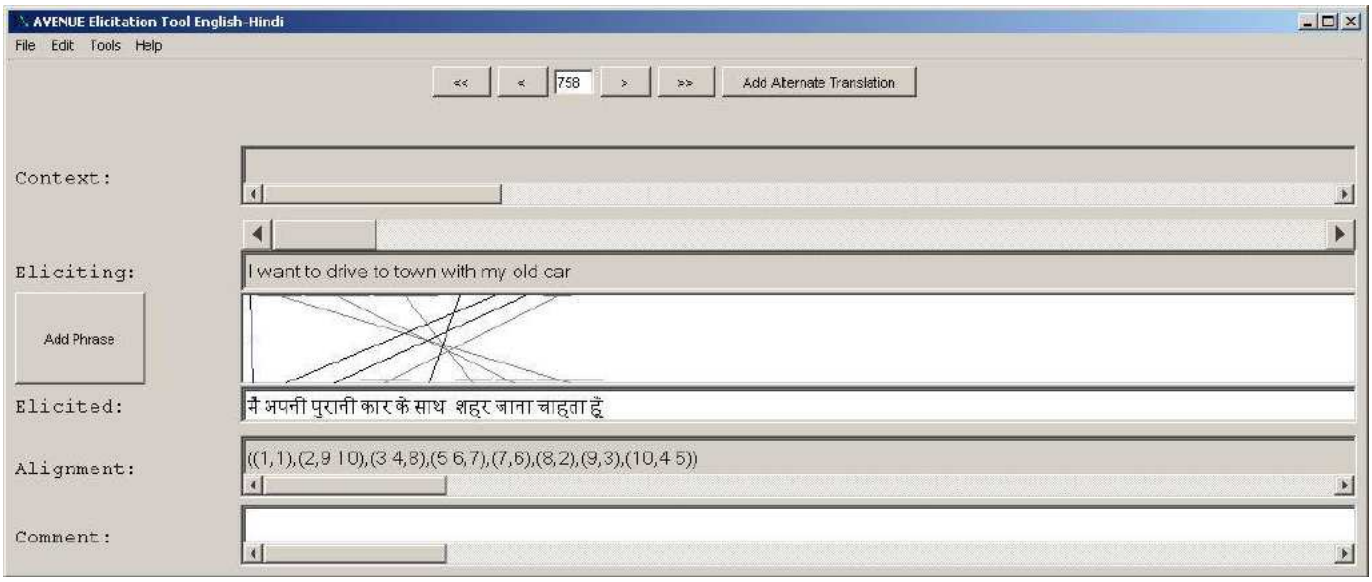

Figure 4.1: Elicitation tool for corpus creation.

are synonyms. As for word alignments, the users are encouraged to align at the word-level as much as possible, and to only resort to many-many alignments when the translation is a fixed expression or an idiom. They are also asked to align those words whose roots correspond to the same concept. If, for example, an English noun translates into a noun and a preposition in Hindi, then the two nouns should be aligned and the preposition should be unaligned, as its root does not carry the meaning of the English noun, but rather carries syntactic information. In the work reported in this thesis, parts of the elicitation corpus (as described below) were translated into Hebrew and Hindi. For Hebrew, three informants translated different parts of the elicitation corpus, and for Hindi, one person served as a bilingual informant. As the translators give not only the word alignment, but also the translation itself, inter-annotator agreement is difficult to assess reliably because of data sparseness. The agreement could reasonably be expected to be high, because the translated sentences and phrases are very short (3.74 words on average).

Rule learning can proceed with an extremely small corpus: about 100 carefully selected phrases and sentences allow us to capture many major phenomena for a language pair. The elicitation corpus can thus be described

as a condensed, but high-variety bilingual training corpus. In the absence of large but uncontrolled corpora, we elicit a small but controlled corpus. The elicitation corpus can be subdivided into two main parts: the functional and the structural part.

The design of the elicitation corpus is not in the scope of this thesis. For coherence and focus, this thesis is aimed at using bilingual data to learn transfer rules. It is however necessary to discuss the structural elicitation corpus so as to clarify the training data that the rule learning module works with for the experiments described in this document. For further details on elicitation, the interested reader may refer to papers such as (Probst & Lavie, 2004; Probst & Levin, 2002), which deals with the corpora, their design, and their goals in greater detail.

4.0.4 Functional Elicitation Corpus

The functional elicitation corpus is designed to cover major linguistic phenomena in typologically diverse languages. The current pilot functional corpus has about 2000 sentences (in English, some of which have been translated into Spanish), though we expect it to grow to at least the number of sentences in (Bouquiaux & Thomas, 1992), which includes around 6,000 to 10,000 sentences, and to ultimately cover most of the phenomena in the Comrie and Smith (Comrie & Smith, 1977) checklist for descriptive grammars. We are currently investigating techniques to create such a corpus automatically by defining comprehensive feature vectors and using a generation grammar that produces sample sentences from the feature structures that can ultimately be used for elicitation.

The current corpus contains phrases and sentences such as the following:

```
The family ate dinner.
The dance amazed the children.
Why are you not going to sleep?
My heart beat.
```

The linguistic features that are emphasized in this part of the corpus are features such as copula, negation, questions, passive, possessives, comparatives, numbers, etc. The most major difference between the functional and the structural corpus is that the functional elicitation corpus does not emphasize structural variation. Rather, it focuses on linguistic features.

The functional elicitation corpus was designed for a multitude of purposes. For instance, it can be used to detect whether certain features exist

in a language (e.g. does a language mark nouns for dual? (Probst et al., 2001)). In the future, we also envision the functional elicitation corpus to be used as input to rule learning. At this stage in development, however, the full functional corpus is incomplete, and is thus not yet ready to be used for this purpose. Since a completed functional corpus would provide an ideal environment for Constraint Learning (as described in chapter 8), we chose to perform a case study where we use one part of the functional corpus, and learn a grammar with constraints from it. This case study deals with Hebrew copulas. The copula corpus contains 287 unique elicitation examples for copula expressions, such as

```
She was the teacher.
They were students.
```

The copula case study is described in section 8.8.of the experiments in this thesis use the structural elicitation corpus.

4.0.5 Structural Training Corpus

The structural part of the elicitation corpus, referred to as the structural training corpus, was designed to cover major structural phenomena. It consists of a few hundred sentences and phrases (120 structures that are exemplified with different sentences and phrases in three corpora), and it covers different compositions of adjective phrases (ADJPs), adverb phrases (ADVPs), noun phrases (NPs), prepositional phrases (PPs), SBARs and sentences (Ss). It also includes a small number of subtypes such as SQ (a question sentence). Examples of each of these types can be found in Figure 4.2. For each of these types of structures, we have collected a number of instances of different composition.

A note on verb phrases (VPs) is in order here: in our system, we do not currently learn VPs. This is because we are mainly interested in constituent who are likely to translate into constituents of the same type. For the above constituent types, this is a reasonable assumption. However, VPs are different in that their make-up often changes significantly during translation. Handling VPs calls for a different approach, such as the one put forth in (Dorr et al., 2004). We discuss possible extensions to our approach to cover VPs in section 10.3.

We began the creation of the structural corpus with a set of parses from the Brown section of the Penn Treebank (Marcus et al., 1995). The Penn Treebank is a large-scale project where different types of text were

1. **ADVP:** Adverb phrase, e.g., 'MAYBE OR MAYBE NOT'
2. **WHADVP:** Question adverb phrase, e.g., 'HOW DISAPPOINTED'
3. **ADJP:** Adjective phrase, e.g., 'EXTREMELY HAPPY'
4. **WHADJP:** Question adjective phrase, e.g., 'SO INTENSE THAT SHE FORGOT'
5. **NP:** Noun phrase, e.g., 'the big book'
6. **WHNP:** Question noun phrase, e.g., 'TEN PICTURES ON THE WALL'
7. **PP:** Prepositional phrase, e.g., 'WHAT UNIVERSITY'
8. **WHPP:** Question prepositional phrase, e.g., 'ON WHICH WERE PAINTED RED AND GREEN APPLES'
9. **SBAR:** Sbar, e.g., 'ESPECIALLY WHEN HE WAS THERE'
10. **SBARQ:** Question Sbar, e.g., 'WHO IS IT'
11. **S:** Sentence, e.g. 'THE BOY ATE AN APPLE'
12. **SQ:** Question sentence, e.g., 'CAN YOU COME'
13. **SINV:** Inverted sentence, e.g., 'ADDED THE SPEAKER'

Figure 4.2: Relevant constituent types.

manually parsed. The parses are often used especially in the statistical parsing community. In that context, they serve to train statistical models of the parse trees that can be used to parse unseen text. In our approach, we made use of the manual parses in a different way. We mapped the Penn Treebank tagset (http://www.computing.dcu.ie/~acahill/tagset.html, 1990). to the tagset used by our system, and then collected statistics over the six types (and their subtypes) in Figure 4.2. Each instance of one of these types is expressed as a *pattern* of expression, where a pattern is the top-node of a partial parse tree, together with the labels of the top-node's immediate children. For example, the following two NPs are considered to be of the same pattern `NP→ ADJP N`.

```
black and white and color photography
Type: NP
Parse:(<NP> (<ADJP> (<ADJP> (ADJ black-1)
                    (CONJ and-2) (ADJ white-3))
                    (CONJ and-4) (<ADJP> (N color-5)))
            (N photography-6))

SL: subsequent developments
Type: NP
Parse:(<NP> (<ADJP> (ADJ subsequent-1))
            (N developments-2))
```

We refer to the pattern of a partial parse tree as its *top-level component sequence*. For each type (ADVP, ADJP, etc.), we collected a large number of different top-level component sequences, together with their frequency of occurrence. This gave us an idea of how different structural types are expressed in English. We then created an elicitation corpus from the most frequent patterns as follows: for each type, we wanted to cover most of the probability mass of different top-level component sequences that can express this type. For those most frequent patterns, we then either extracted or designed a generic, simple example for elicitation. Referring back to the two NP examples from above, the first example would be a better elicitation sentence, because it is simple. This will make it more likely that the sentence is actually translated into a similar structure in the other language, or else that the structure in the other language can be robustly detected automatically. The second sentence allows enough room for translation variation that the task of the rule learning system would be made more difficult. In a post-editing step, we then changed a number of lexical items so as to remain as culturally

unbiased as possible. This was done in part because lexical selection can cause sentences to be translated into completely different structures, and in part because this corpus is designed to be used for various language pairs, and culturally biased lexical selection can lead to sentences that are simply not translatable into many languages. During post-editing, we also eliminated a number of structures that should be elicited in the functional part of the elicitation corpus, because it is known to be expressed differently in different languages, and thus deserves special in-depth treatment. One such example is the partitive, e.g., 'SOME OF THE CHILDREN'. Below we list several examples of elicitation sentences from the structural corpus:

```
The election was conducted.
the widespread interest in the election
the city executive committee
David can barely walk.
```

In (Probst & Lavie, 2004) we describe a different approach to overcome the problems that lexical selection can cause. There, we make several copies of the elicitation corpus, where in each copy the lexical items in the elicitation examples are changed. Each corpus, however, has the exact same structures. We show there that lexical selection does indeed play a role in what types of structures are elicited, but that this problem can be overcome with only a small number of corpus copies.

While this is an interesting and promising approach, it is not the one taken in this thesis. Rather, in the results reported here, we put the learning system in the extreme situation of having to learn from only one copy of a very small corpus. The goal was to determine what we can learn under those severe circumstances and the algorithms were optimized for this task. If we can learn meaningful rules and improve translation quality, we show that the approach put forth in this thesis can really be applied to language pairs where we can obtain only very little data.

The structural elicitation corpus consisting of 120 sentences and phrases was translated and aligned for both Hebrew and Hindi. Unless otherwise noted, all learned grammars that are described in this document were learned from the structural corpus. Some bilingual examples between English and Hebrew from this corpus can be found below. Together with the sentence pairs, we list their English parses and the user-specified alignments. The following section we describe the full training data format for the rule learner.

```
English: in the forest
```

```
Hebrew: B H I&R
Alignment: ((1,1),(2,2),(3,3))
C-Structure:(<PP> (PREP in-1)(<NP> (DET the-2)(N forest-3)))

English: quickly
TL: B MHIRWT
Alignment: ((1,1),(1,2))
C-Structure:(<ADVP> (ADV quickly-1))

English: the boy ate the apple
Hebrew: H ILD AKL AT H TPWX
Alignment: ((1,1),(2,2),(3,3),(4,5),(5,6))
C-Structure:(<S> (<NP> (DET the-1)(N boy-2))(<VP> (V ate-3)
    (<NP> (DET the-4)(N apple-5))))

English: a dispute with the school board
Hebrew: SKSWK &M W&D BIT H SPR
Alignment: ((2,1),(3,2),(4,5),(5,4),(5,6),(6,3))
C-Structure:(<NP> (<NP> (DET a-1)(N dispute-2))
    (<PP> (PREP with-3)(<NP> (DET the-4)(N school-5)
    (N board-6))))

English: old
Hebrew: I$N
Alignment: ((1,1))
C-Structure:(<ADJP> (ADJ old-1))

English: where our interests lie and what we must do
Hebrew: HIKN $ H AIN@RSIM $LNW NMCAIM W MH $ ANXNW CRIKIM
    L&$WT
Alignment: ((1,1),(2,5),(3,4),(4,6),(5,7),(6,8),(7,10),(8,11),
    (9,12))
C-Structure:(<SBAR> (<SBAR> (<WHADVP> (WH where-1))
    (<S> (<NP> (POSS our-2)(N interests-3))(<VP> (V lie-4))))
    (CONJ and-5)(<SBAR> (<WHNP> (SUBORD what-6))
    (<S> (<NP> (PRO we-7))(<AUX> (AUX must-8))
    (<VP (V do-9)))))

English: aware that the money was gone
Hebrew: MWD& $ H KSP N&LM
```

```
Alignment: ((1,1),(2,2),(3,3),(4,4),(5,5),(6,5))
C-Structure:(<ADJP> (ADJ aware-1)(<SBAR> (SUBORD that-2)
    (<S> (<NP> (DET the-3)(N money-4))(<AUX> (V was-5))
    (<ADJP> (ADJ gone-6)))))

English: because he was hungry
Hebrew: KI HWA HIH R&B
Alignment: ((1,1),(2,2),(3,3),(4,4))
C-Structure:(<SBAR> (PREP because-1)(<S> (<NP> (PRO he-2))
    (<VP> (V was-3)(<ADJP> (ADJ hungry-4)))))
```

4.0.6 Training Data Format

During elicitation, the bilingual informants provide the translation of an elicitation example as well as word-level alignments.

In addition, the rule learner is given the English parse of each training example. In the case of the structural training corpus, the parse is the parse from the Penn Treebank (with possible small changes if the lexical items were changed from the original sentence in the Penn Treebank to the sentence used in the structural corpus). For the functional corpus, we used the Charniak parser (Charniak, 2000). As all the training examples in the functional corpus are very short and structurally simple sentences, the Charniak parser yields very reliable results.

In addition to a translation pair, word alignments, and the English parse, each training example also contains *type* information as well as a *co-embedding score*. The type is the label of the top-node of the English parse of each example. The co-embedding score essentially captures the depth of the parse tree. It will be described in more detail in chapter 7, as it is used during Compositionality Learning. Type and co-embedding scores can be obtained off-line before learning. This results in improved efficiency of the rule learner.

A complete training example then looks as follows:

```
TL: in the forest
SL: B H I&R
Alignment:((1,1),(2,2),(3,3))
Type: PP
CoEmbeddingScore: 3
C-Structure:(<PP> (PREP in-1)(<NP> (DET the-2)(N forest-3)))
```

4.0.7 A Note on Uncontrolled Corpora

The rule learning techniques developed in this thesis are not aimed specifically at the controlled elicitation corpora described in the previous sections. In fact, the rule learning module could be run just as easily on any bilingual data that is available, with automatically assigned word alignments.

Uncontrolled corpora have the advantage of being more widely available. It is often possible to find a bilingual publication for a given language pair, for example an online newspaper. Then automated techniques for sentence and word alignment can be applied to this uncontrolled corpus, yielding the type of corpus that is needed for the learning algorithms described here. On the other hand, controlled corpora have the advantage of that 1) they can easily be developed by bilingual speakers if no uncontrolled corpus is available, and 2) the researchers have more control over what the training corpus contains. The latter advantage allows us to compact the corpus by making it diverse in the types of structures that it contains.

We have developed some techniques to learn under a variety of circumstances, such as from uncontrolled corpora and from automatically induced SL morphology modules (Probst, 2003). However, this is not the main line of this research. The goal of this thesis was to investigate what can be learned under a miserly data scenario, for example from only the structural elicitation corpus of 120 sentences. As mentioned above, if we can learn meaningful grammars from a corpus this small, we can make a strong case that our approach can be applied to language pairs where very little data is available. We show in this thesis that this is the case.

Learning from uncontrolled corpora is a very exciting and promising direction that this research could take in the future. It is however an equally big project to the one described in this thesis. In particular, as we show in chapter 9, the algorithms that perform well under the miserly data scenario are not the same ones that perform well when larger amounts of training data are given. This indicates that for a different scenario, i.e. large datasets, the problem must be approached in a slightly different way. For the purpose of this thesis, this problem is considered to be out of scope.

Chapter 5

Evaluation Methodology

Throughout the thesis, the learned grammars are evaluated in two ways: first by a discussion of the learned rules and their application, and second in a 'task-based' evaluation where the rules are used in the context of a translation system. The discussion of the learned rules is both useful and important, as the system learns rules that are human-readable and can thus be inspected for their soundness. For each module, and each substantial learning phase, we thus discuss the learned rules and their strengths and weaknesses. We also demonstrate how the learned rules in a specific module accomplish the goal of this module. For example, if a module is designed to solve the problem of pro-drop, we demonstrate how the learned rules capture pro-drop and improve translation quality.

In the task-based evaluation, the grammar is used with the transfer engine to produce a lattice, and the decoder is used to produce a final translation. The translations are evaluated with automated evaluation techniques.

The first metric that we use is BLEU (Papineni et al., 2001). BLEU is a metric that is routinely used in the MT community to automatically assess translation quality. It is based on matching a hypothesis translation (i.e. the output of an MT system) to one or more reference translations that were produced by a human. BLEU counts how many unigrams, bigrams, ... n-grams (where n is usually set to 4) in the hypothesis translation match one or more reference translation. For each n-gram length, a score is assigned. The BLEU metric then takes the geometric mean of the scores assigned to all n-gram lengths. This is however not enough: the MT system could propose very long translations and thus stand a better chance of matching many n-grams. In order to overcome this potential problem, BLEU introduces the notion of a lengthlimit, where translation hypotheses are penalized if their

length is different from the average reference translation length.

As BLEU is not normalized and increases with the number of sentences in the test set and with the number of reference translations, it is best used as a comparison metric to assess the difference of two systems. This is how it is used in this thesis. In particular, we compare the performance of the system without learned rules to the system with learned rules.

One drawback of BLEU is that if the number of matched higher-order n-grams is small, the overall score will be low, regardless of how many lower-order n-grams are used. In particular, if for example no n-grams of a specific length (e.g. no 4-grams) are matched, the overall score will be 0. In order to overcome this problem, a modified version of the BLEU metric, the modified BLEU (referred to as ModBLEU in this document) has been proposed by Zhang (Zhang & Vogel, 2004). Instead of the geometric mean, they use the arithmetic mean, which corrects the problem. We report this metric in this document for completeness. In our experience, however, its behavior is usually very similar to the BLEU metric if there are no zero matches of n-gram lengths.

The third metric used in this thesis is METEOR. The METEOR metric, proposed in (Lavie et al., 2004) also matches unigrams, bigrams, ..., n-grams to reference translations. However, instead of a length penalty, it counts the number of reorderings that would be necessary to get from the hypothesis translation to the reference translation.

It is important to note that BLEU and ModBLEU put a stronger emphasis on precision than on recall, whereas METEOR puts more emphasis on recall. This basic difference is consistently observed in our results. For example, whenever an algorithm is designed to improve recall while sacrificing some precision, the METEOR score generally increases, while the BLEU and ModBLEU scores decrease.

In addition to the translation evaluation metrics, we also use a Lattice Evaluation, which abstracts away from the decoder: it evaluates the lattice produced by the transfer engine by comparing it to the reference translation(s) much in same way as BLEU and METEOR do. The lattice evaluation compares the translation chunk of each lattice arc against the reference translation. It first counts the number of unigrams, then the number of bigrams, etc. up to a specified maximum n-gram size. For each match, the arc's score is increased. The lattice scoring also takes into account the length of the arc, and how many times each of the matched n-grams was already matched in a given sentence. If an arc contains an n-gram match that was already proposed by another arc, then the addition to the arc score is discounted based on how many times the matched n-gram was already

observed. In this way, an arc receives a boost in score for each matched n-gram. However, frequently matched n-grams are not rewarded as highly, because an arc that contains a frequently matched n-gram adds very little to the quality of the lattice. Multiple reference translations are handled by computing the score for an arc for each reference translation, and then retaining only the maximum value. Since the evaluation method is not directly part of the thesis, more details can be found in the appendix under section A. In the work reported here, the lattice evaluation metric is used to assess the quality of individual rules. We can measure how well rules perform (independently of the decoder) by evaluating the quality of the arcs that were produced using this rule. More details on rule scoring can be found in section 9.5.

We report results for each individual learning phase and each training set, and compare the results. In addition to comparing different grammars, we also compare the learned grammars to a baseline and a manually written grammar. The baseline is a translation without a grammar, but only with the statistical decoder and the lexicon. The manually written grammar is a small Hebrew→English grammar that was designed by a human expert in the course of a few weeks. While by no means comprehensive, this grammar provides a good basis for comparison. For Hindi→English translation, we only compare to the baseline without a grammar.

The experiments were performed on a test set of 26 parallel sentences of newspaper style text with one reference translation (test set 1). In addition, we evaluate the algorithms on an test set of 62 sentences, again newspaper test, but with two reference translations (test set 2). Test test 1 was used as a development set for the manually written grammar. For this reason, the manual grammar performs better on test set 1 than on test set 2. For automatic rule learning, both test set 1 and test set 2 are unseen test data. We further evaluated on a test suite that targets specific syntactic phenomena of interest, for instance reordering of adjectives and nouns between Hebrew and English. The test suite has 138 sentences and one reference translation. The test suite and the evaluation on it will be discussed in section 9.6.

The run-time system is generally used in a mode that allows the user to specify the maximum arc length of partial translations as discussed in section 3.3. For example, a lengthlimit of 6 means that the maximum number of source indices (i.e. components) that can be spanned by any one arc is 6. While this is suboptimal in the sense that we do not obtain complete lattices, the lengthlimit is a very useful tool in practice. It is set in order to combat the often very large number of possible arcs. The decoder chooses between the possible partial translations, but when there are sufficiently

many partial translations, the transfer engine is not able to produce all of them. Unless otherwise specified, we report results throughout for a length-limit of 6. In section 9.7, we compare the performance of grammars under different lengthlimits.

In addition to BLEU, ModBLEU, and METEOR scores, we report throughout the thesis confidence intervals and p-values that can be associated with the raw scores. In the case of BLEU and ModBLEU, no sentence-level score is available. For this reason, it is not completely straightforward to measure statistical significance. However, it would be useful to report a measure of statistical significance between the baseline and the various learned grammars. For this reason, we follow the method described in (Zhang & Vogel, 2004), and used the software provided with it. The authors apply the technique of bootstrapping in order to produce enough information to report confidence intervals. The interested reader should refer to the paper for more details. We explain the idea here only briefly: For BLEU and ModBLEU, the bootstrap proceeds as follows: suppose that the test set consists of x sentences. Then the bootstrap randomly draws from the x sentences x times *with* replacement. In other words, it randomly creates a new artificial test set, also consisting of x sentences, but in the new test set some sentences may be repeated. For this test set, it then computes the BLEU and ModBLEU scores both for the baseline and for the system that uses the learned grammar, and takes the difference between the scores. This process is repeated y times, where the y used in the experiments in this thesis was set to 10000. This yields 10000 data points of the difference between the baseline and the system with the learned grammar. Finally, the bootstrapping module reports a 95% confidence score on this difference. In other words, if 0 is not included in this confidence interval, we can say with 95% confidence that the baseline and the system with the learned grammar are different. Often in our results, 0 is included in this interval, but only barely, leading us to conclude that the system with the learned grammar are different with *close to* 95% confidence.

Statistical significance for METEOR scores are easier to compute. Here, we report the p-value of a one-tailed t-test, again between the baseline and the system with the learned grammar. The p-value is the lowest confidence level at which we can reject the hypothesis that the baseline and the system with the learned grammar are *not* different. In other words, a p-value of 0.05 indicates that with 95% confidence, the two systems *are* different.

In the tables that report the confidence intervals and p-values, we only give the name (or description) of the learned grammar. The reader should note that we always report confidence intervals on the *difference* between the

system with the learned grammar and the baseline. Similarly, the p-values always indicate the level of statistical significance of a one-tailed t-test on the *difference* between the system with the learned grammar and the baseline.

It should be noted that in particular for the Hebrew test sets, a statistically significant result is difficult to accomplish, because the test sets are small (26 sentences and 62 sentences, respectively). On the other hand, a statistically significant result on such a test set is a very strong result.

Chapter 6

Seed Generation

6.1 Introduction

Seed Generation is the process of building transfer rules from the training data that closely reflect the sentences that they were derived from. This is done using the parses, the dictionary, and a morphology module for the minor language SL.

The goal of this module is to produce flat rules. To clarify this concept, consider the following two rules, NP,1 and NP,2:

```
{NP,1}
;;SL: &NIIN RXB B H BXIRWT
;;TL: WIDESPREAD INTEREST IN THE ELECTION
NP::NP [N ADJ PREP DET N] -> [ADJ N PREP DET N]
(
(X1::Y2)
(X2::Y1)
(X3::Y3)
(X4::Y4)
(X5::Y5)
)

{NP,2}
;;SL: &NIIN RXB B H BXIRWT
;;TL: WIDESPREAD INTEREST IN THE ELECTION
NP::NP [N ADJ PP] -> [ADJ N PP]
(
(X1::Y2)
```

```
(X2::Y1)
(X3::Y3)
)
```

The first rule, NP,1, is an example of a *flat* rule. Its component sequences contain only POS labels, but no constituent labels. The second rule, NP,2, on the other hand, contains a generalization to the constituent level, represented by the PP labels in both component sequences. The second rule can combine with any rule whose type is PP::PP. The first rule, on the other hand, cannot combine with any other grammar rules and is therefore considered a 'flat' rule. The goal of Seed Generation is to produce such flat rules. Flat rules and their relationship to Seed Generation can be defined as follows. Please refer to Figure 3.3 for a reminder of essential rule parts.

A flat rule is a fully functional transfer rule whose component sequences contain only lexical items or part of speech labels, but no constituent labels. Seed Generation produces exclusively flat rules.

Constituent labels such as PPs, which allow the rules to combine, are introduced during Compositionality Learning. Flat seed rules, on the other hand, can only apply individually. In addition, seed rules are generally highly lexicalized: with few exceptions, any word that is not aligned one-one is left lexicalized during Seed Generation. Compositionality learning overcomes this one-one restriction in some cases.

Seed generation, like all other learning modules, learn rules only for a set of *relevant types*. The constituent types that are handled in our system are listed in Figure 4.2.

Other non-terminals, in particular verb phrases (VPs), are very difficult to capture in transfer rules, because they are more likely to exhibit structural mismatches. Since our rules are learned completely automatically, excluding VPs from our choice of relevant types is merely a conservative precaution. This does not mean that verbs are not an integral part of the learned rules, but no sequence of parts of speech and/or non-terminals is ever generalized to a VP, and no rule are learned that are of type VP::VP. A possible extension to our work to include VP handling is discussed in section 10.3.

6.2 Description of Learning Algorithm

Seed Generation processes each training example in turn. For each, it constructs a flat transfer rule called a *seed rule*, if certain conditions are met. The seed rule is a complete transfer rule in format: it contains SL and TL type information, component sequences, and alignments. In other words, it is a fully functional transfer rule.

For each training example, the task of the Seed Generation module is to produce the following rule parts:

- SL type information
- TL type information
- SL component sequence
- TL component sequence
- word-level alignments

In Figure 6.1, we present the Seed Generation algorithm in pseudocode. While it is not possible to include all details in the pseudocode listing, it can be useful for giving an overview picture.

The TL type information is obtained from the parse, it is simply the label of the top node. It is important to remember that at Seed Generation time, each bilingual training example is fully annotated, including the English syntactic parse. For this reason, the English parse for each training example is readily available to the Seed Generation algorithm. For more details on what information is available for each bilingual training sentence pair, please refer to chapter 4.

The SL type information is assumed to be always the same as for the major language, TL, so that the SL type information in each seed rule is also the TL parse's top node. This approach stems from the underlying assumption that elicitation targets those sentence components that are likely to transfer into the same kind of component, e.g., noun phrases are assumed to translate into noun phrases.

The TL component sequence can essentially be obtained from the English parse that is given with the training example: for each word in the training example's TL sentence or phrase, the part of speech of this word can be looked up in the parse. One restriction that is put on the component sequences is that only words that are aligned one-one can be represented by their POS in the component sequence. By contrast, not one-one aligned

Seed Generation

For all training examples

For all 1-1 aligned word pairs $w_{i,SL}, w_{j,TL}$

Get the TL POS tag $POS_{w_{j,TL}}$ for $w_{j,TL}$ from the parse

Get the POS tag $POS_{w_{i,SL}}$ for $w_{i,SL}$ if possible:

If there exists an entry in the dictionary for the pair ‘‘$w_{i,SL}$’’→‘‘$w_{j,TL}$’’

If the POS combination for this entry is $POS_{w_{j,TL}}$ :: $POS_{w_{j,TL}}$, then let $POS_{w_{i,TL}}$ be $POS_{w_{j,TL}}$

Else leave $w_{i,SL}$ and $w_{j,TL}$ lexicalized

Else get a set $POS_{1..k,w_{i,SL}}$ of all possible POS tags for $w_{i,SL}$ from the SL morphology

If $POS_{w_{j,TL}} \in POS_{1..k,w_{i,SL}}$, then let $POS_{w_{i,TL}}$ be $POS_{w_{j,TL}}$

Else if $POS_{1..k,w_{i,SL}}$=$\emptyset$, then let $POS_{w_{i,TL}}$ be $POS_{w_{j,TL}}$

Else leave $w_{i,SL}$ and $w_{j,TL}$ lexicalized

For all other words, leave the words lexicalized

For each SL word $w_{i,SL}$ that is left lexicalized, determine the most likely root:

Get a set $root_{1..k,w_{i,SL}}$ of all possible roots for $w_{i,SL}$ from the SL morphology

For each aligned TL word $w_{j,TL}$ and each $root_{w_{i,SL}} \in root_{1..k,w_{i,SL}}$, look up the combination ‘‘$root_{w_{i,SL}}$’’→‘‘$w_{j,TL}$’’ in the dictionary

If a combination exists, then $root_{w_{i,SL}}$ remains in the set $root_{1..k,w_{i,SL}}$

Else remove $root_{w_{i,SL}}$ from $root_{1..k,w_{i,SL}}$

If exactly one element remains in $root_{1..k,w_{i,SL}}$, then this is the root

Else if the original set $root_{1..k,w_{i,SL}}$ as returned from morphology has exactly one element, this element is the root

Else the word itself $w_{i,SL}$ is considered the root

Produce a fully functional transfer rule from the POS tags, the words that remain lexicalized, and the alignments

Figure 6.1: Pseudocode for Seed Generation.

words are left lexicalized, i.e. the lexical item from the translation sentence remains in the component sequence. This was seen earlier in the sample transfer rule in Figure 3.3, where two instances 'H' and 'THE' remained lexicalized because they were not aligned one-one.

SL component sequences are obtained similarly to their TL counterparts. Any one-one aligned word is replaced by a POS label if certain conditions hold. As the rule learning system works without an SL parse, the bilingual dictionary is used to obtain a better estimate of what the POS of a given SL word should be. For all one-one aligned words, the TL-SL word combination is first looked up in the dictionary. Similarly, if the TL-SL word combination is found in the dictionary and the dictionary entry's TL POS is the same as the TL word's POS in the parse, the words are raised to the POS level. However, if the SL part of speech is not the same, then both the TL word and the SL word are left lexicalized. If there is no entry in the lexicon for this combination, the SL morphology module is used to return a list of possible parts of speech for the given word. If no possible parts of speech are returned, it is assumed (in the absence of any other information) that the SL POS label is the same as the POS of the corresponding TL word. Both the TL and SL word will be replaced by their parts of speech in the learned seed rule.

If one of the possible parts of speech returned by the morphology module is the same as the corresponding TL word's POS, then both the TL and the SL word are generalized to the POS level, and this POS will be entered into the component sequence.

If a word is not aligned one-one, it must remain lexicalized. In the Hebrew→English system, the dictionary is in full (i.e. inflected) form for English, but in root form for Hebrew. Therefore it is necessary that the rules contain Hebrew lexical items in their root form if the words are lexicalized. The root form can be obtained as follows: the Hebrew word is analyzed by the Hebrew morphology module, which returns a list of possible roots. Each root and all possible English translations from the training example are then looked up in the dictionary. If there exists an entry for such a combination, then the proposed root is returned as the root for the given SL word. If no such entry exists, but the morphology module returns exactly one possible root, then this root is returned. Otherwise, no root is returned. If a root is found, it is entered into the SL component sequence instead of the original word. If not, the original Hebrew lexical item is entered into the SL component sequence.

As was mentioned above, the Seed Generation module produces completely 'flat' rules. This is accomplished by allowing the component se-

quences to contain only literals and parts of speech, i.e. only components that refer to at most one word in the training examples. Components that can span more than one word, and can be filled by other grammar rules, are learned during Compositionality.

The Seed Generation algorithm contains some technical details that could obscure the big picture. For this reason, we summarize below the essential goals and assumptions of Seed Generation. The three most important details to remember about Seed Generation are the following:

1. **Seed Generation produces flat rules.**
2. **Not one-one aligned words remain lexicalized in the component sequences.**
3. **The SL type information is assumed to be the same as the TL type information.**

6.3 Results

6.3.1 Discussion of Learned Rules

We will now discuss several rules for Hebrew→English translation that were learned using the Seed Generation module. All rules were learned from the structural elicitation corpus, which consists of 120 sentences and phrases. For details on the training corpus, refer back to chapter 4.

```
;;SL: $MX MAWD &L H XD$WT
;;TL: VERY HAPPY ABOUT THE NEWS
;;C-Structure:(<ADJP> (<ADVP> (ADV very-1))(ADJ happy-2)
    (<PP> (PREP about-3)(<NP> (DET the-4)(N news-5))))
ADJP::ADJP [ADJ ADV PREP DET N] -> [ADV ADJ PREP DET N]
(
(X1::Y2)
(X2::Y1)
(X3::Y3)
(X4::Y4)
(X5::Y5)
)
```

This rule demonstrates well what kind of structures Seed Generation can capture. It was learned that the English pre-adjectival adverb appears after the adjective in Hebrew. The position of the prepositional phrase is

the same for both languages. Note that this rule does not contain any lexicalized items, as all words are aligned one-one in the training example, and there is evidence that the parts of speech for the Hebrew words are the same as for their aligned English counterparts. A rule such as this first example will readily apply to unseen text. It illustrates generalization over the training data up to the POS level. Any lexicon entries of these parts of speech can now fill the slots in the rule. For example, any lexicon entry of type ADJ::ADJ can now fill the slots X1 and Y2, which are aligned to each other and of type ADJ.

The following is an example of a less general, more lexicalized rule:

```
;;SL: H QBWCH H XD$H $LW
;;TL: HIS NEW TEAM
;;C-Structure:(<NP> (POSS his-1)(ADJ new-2)(N team-3))
NP::NP ["H" N "H" ADJ POSS] -> [POSS ADJ N]
(
(X2::Y3)
(X4::Y2)
(X5::Y1)
)
```

This rule expresses how simple possessives, in conjunction with an adjective, are realized in Hebrew. It was learned that while an English NP with a possessive, as in the example, generally does not mark for definiteness, the Hebrew definiteness marker 'H' appears on both the noun and the adjective. Both instances remained lexicalized because they are not aligned to any English words. It should also be noted that the entire phrase is completely reordered in Hebrew.

The following is another automatically learned rule:

```
;;SL: KL KK XZQ $ HIA $KXH
;;TL: SO INTENSE THAT SHE FORGOT
;;C-Structure:(<ADJP> (<ADJP> (<ADVP> (ADV so-1))(ADJ intense-2))
(<SBAR> (SUBORD that-3)(<S> (<NP> (PRO she-4))(<VP> (V forgot-5)))))
ADJP::ADJP ["KL" "KK" ADJ SUBORD PRO V] -> ["SO" ADJ SUBORD PRO V]
(
;(X1::Y1)
;(X2::Y1)
(X3::Y2)
(X4::Y3)
```

```
(X5::Y4)
(X6::Y5)
)
```

This is an example of a partially lexicalized rule that nevertheless captures an interesting Hebrew expression and its English counterpart. The English intensifier 'SO' is expressed in Hebrew as the two words 'KL' 'KK'. The words remained lexicalized during learning because they were not aligned one-one. This rule, albeit interesting, will apply only to few test examples because of its partial lexicalization. Note that the transfer rule contains, in commented-out form, the word alignments that were specified by the user between 'KL' 'KK' and 'SO'. These alignments are merely retained for the human inspector; they are commented out automatically by the rule learner and ignored by the transfer engine.

Finally, we present an example of reordering of noun compounds between Hebrew and English.

```
;;SL: TKNIT H @IPWL H HTNDBWTIT
;;TL: THE VOLUNTARY CARE PLAN
;;C-Structure:(<NP> (DET the-1)(<ADJP> (ADJ voluntary-2))
    (N care-3)(N plan-4))
NP::NP [N "H" N "H" ADJ] -> ["THE" ADJ N N]
(
(X1::Y4)
;(X2::Y1)
(X3::Y3)
;(X4::Y1)
(X5::Y2)
```

As previously, the Hebrew definite determiner 'H' appears more than once. In Hebrew, noun compounds mark definiteness only on the second noun. The determiner is repeated on the adjective. As the alignments indicate, the word order is essentially reversed when translating from Hebrew into English.

6.3.2 Automatic Evaluation Results

Seed Generation is the first of three phases of rule learning. The rule learning module can be run only for Seed Generation, producing flat seed rules. It is interesting to evaluate these rules without running the subsequent two

learning phases. The seed rules capture important phenomena of word reorderings, insertions, deletions, etc. The two following learning phases turn the flat seed rules into more generally applicable rules, but even the flat seed rules themselves capture a variety of translation phenomena.

We ran the rule learner for Seed Generation only on the structural elicitation corpus of 120 sentences and phrases. For the learned grammar, we computed several automated evaluation scores. We report here the BLEU score, the ModBLEU score, as well as the METEOR score. Here and for the remainder of the document, we compare against a baseline translation without a grammar and a translation with a hand-written grammar, as described in chapter 5.

Grammar	**BLEU**	**ModBLEU**	**METEOR**
No Grammar	0.0255	0.0910	0.2681
Manual Grammar	0.0713	0.1209	0.3204
Learned Grammar (SeedGen)	0.0281	0.0969	0.2786

Table 6.1: Seed Generation only evaluation results on test set 1.

Comparison	**BLEU**	**ModBLEU**	**METEOR**
SeedGen	[-0.0073,0.0018]	[-0.0149,0.0026]	p=0.112

Table 6.2: Seed Generation only confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline.

Table 6.1 presents the results on test set 1 for the baseline without a grammar, manually written grammar, and the grammars that were learned from the structural corpus. The p-value and confidence intervals are given in Table 6.2.

Although the results are not highly statistically significant (which is in part due to the small dataset of 26 sentences), it can be seen from the automated evaluation results that considerable improvement of translation quality can be gained from the flat seed rules alone. This indicates that the seed rules capture the kinds of structural phenomena that are targeted during learning. It also indicates that the rules can generalize beyond the training sentences they were inferred from, and can be used to translate unseen examples. In subsequent chapters, we will improve upon this first learning phase. Seed generation is a learning strategy that captures the

training data in rule format, but it does not generalize as well over the training data as Compositionality learning (cf. chapter 7). Generalization is important especially when the training corpus is very small, such as our corpus of 120 sentences and phrases.

We note again that for a test set of 26 sentences, a statistically significant result is difficult to accomplish.

Under the above comparison of the baseline, manual grammar, and the rules learned with Seed Generation, the manual grammar outperforms the learned grammar noticeably. This is in part because test set 1 was used as a development set for the manual grammar. The goal of the following learning phases is then to improve upon the performance of the seed rules, and to get as close as possible or beyond the performance of the manual grammar. We will in fact show that under certain conditions, the learned grammar will outperform the manual grammar.

Chapter 7

Structural Learning

7.1 Introduction

The previous chapter described the Seed Generation algorithm, during which a set of rules are learned that closely reflect the training data. In this and the following chapter, we shift our attention to learning more complex rules, i.e. rules that generalize further over the training data, and that capture higher-level structure as well as context. In order to make clear the difference between the flat rules produced during Seed Generation and the more general rule produced during Compositionality, consider first the following training example and the corresponding seed rule as generated by the learning module. Note again that the English (i.e. TL) parse is given with the training example, and is used during rule learning.

```
TL: a year later they returned
SL: $NH MAWXR IWTR HM $BW
Alignment: ((2,1),(3,2),(3,3),(4,4),(5,5))
Type: S
CoEmbeddingScore: 4
C-Structure:
 (<S> (<ADVP> (<NP> (DET a-1)(N year-2))
              (ADV later-3))
      (<S> (<NP> (PRO they-4))
           (<VP> (V returned-5))))

;;SL: $NH MAWXR IWTR HM $BW
;;TL: A YEAR LATER THEY RETURNED
S::S [N "MAWXR" "IWTR" PRO V] -> ["A" N "LATER" PRO V]
```

```
(
(X1::Y2)
;(X2::Y3)
;(X3::Y3)
(X4::Y4)
(X5::Y5)
)
```

The seed rule is flat, as desired by the Seed Generation algorithm. In order for this rule to scale to a larger number of test sentences, several generalizations of the seed rule are possible:

```
;;SL: $NH MAWXR IWTR HM $BW
;;TL: A YEAR LATER THEY RETURNED
S::S [N "MAWXR" "IWTR" PRO V] -> ["A" N "LATER" PRO V]
(
(X1::Y2)
;(X2::Y1)
;(X3::Y3)
(X4::Y4)
(X5::Y5)
)

;;SL: $NH MAWXR IWTR HM $BW
;;TL: A YEAR LATER THEY RETURNED
S::S [N "MAWXR" "IWTR" NP V] -> ["A" N "LATER" NP V]
(
(X1::Y2)
;(X2::Y3)
;(X3::Y3)
(X4::Y4)
(X5::Y5)
)

;;SL: $NH MAWXR IWTR HM $BW
;;TL: A YEAR LATER THEY RETURNED
S::S [ADVP PRO V] -> [ADVP PRO V]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
```

```
)

;;SL: $NH MAWXR IWTR HM $BW
;;TL: A YEAR LATER THEY RETURNED
S::S [N "MAWXR" "IWTR" S] -> ["A" N "LATER" S]
(
(X1::Y2)
;(X2::Y3)
;(X3::Y3)
(X4::Y4)
)

;;SL: $NH MAWXR IWTR HM $BW
;;TL: A YEAR LATER THEY RETURNED
S::S [ADVP S] -> [ADVP S]
(
(X1::Y1)
(X2::Y2)
)
```

This chapter focuses on learning more general rules such as the ones above. More specifically, it targets the learning of appropriate component sequences. These sequences, together with the type information, represent the structural portion of the rules. If as in the above rules the component sequences contain constituent labels, then they capture a hierarchical structural transfer, i.e. the transfer of structural SL trees to TL trees, because the constituent labels in the component sequences allow the rules to combine with each other. In this sense, the rules are now compositional. Why are structural transfers important? First, they allow us to reorder not only words, but also entire constituents. Second, they also allow us to capture in part the structure of the minor language SL. As mentioned in the Related Work section (chapter 2), a very important distinction between our work and a body of recent related work is that in our framework, we do not assume the availability of an SL parser. This makes the learning of structures all the more challenging. This chapter provides a discussion of learning structure with only a parser for the major language: it discusses what is learnable, and proposes learning algorithms for those structures that can be learned in our framework.

An important observation is that the generalized rules above are of varying degrees of generality. More specifically, the last rule is the most general

rule, which means that it will apply to a large number of unseen examples, while at the same time maximally generalizing over the specific training example. One of the most basic problems that we are facing when learning transfer rules is the trade-off between scaling to unseen examples and overly general rules that apply in many contexts where they should not apply. The goal of Compositionality, as described in this chapter, is essentially to create very general rules that will apply to a variety of new contexts. By contrast, the unification constraints serve to limit the applicability and/or output of the rules. In this way, Compositionality and Constraint Learning together serve to strike an appropriate balance, as supported by the training data.

7.2 Taxonomy of Structural Transfers

In this section, we discuss the theory of what structures can exist, and what structures are learnable. There are several factors to consider:

1. The space of possible transfers
2. The space defined by the rule formalism
3. The space defined by the learning algorithm

The following sections will provide discussions of each of these spaces. In particular, we will discuss the advantages and disadvantages of the rule formalism. We will then derive an argument of what the learning algorithms address and why. Our goal is to at least address in discussion all important types of transfers; in the case of structural transfers that are not learnable by our algorithms, we give a discussion of why this is the case. Some issues are left for future work. The goal of this section is to give a comprehensive discussion of the structural learning problem. Before discussing the three spaces within which the learning system resides, we give a listing of the specific constraints on our system that do not influence the generality of the argument following it.

7.2.1 System Constraints

In this section, we lay out a set of constraints under which our system must work. The constraints are meant to not influence the generality of the approach. Instead, they make the discussion feasible by making it concrete.

We apply the following constraints:

1. Transfers are applied at most at the sentence level. Each transfer (rule) captures a transformation of a phrasal instance or a complete sentence, but not a paragraph.

2. The transfer rules are designed to be unidirectional from the minor language (SL) to the major language (TL).

3. The rules must be learned under the resource constraints described in previous chapters, in particular the availability of a parser *only* for the major language. Due to these restrictions, the structural part of the rules are highly dependent on the TL parse information.

7.2.2 General Space of Possible Transfers

The general space of structural transfers can be regarded as any transfer between the two languages. The most specific transfer is between two completely lexicalized sentences. This is not a structural transfer, because it does not provide any generalization power: no other (partial) sentences can be translated with such a transfer. Further, the transfer is flat, thus does not provide any hierarchy. Example-based machine translation operates mostly at this level.

In this work, we aim at inferring more structural transfers, meaning that they will 1) generalize to unseen examples, i.e. abstract away from the lexical level as much as possible, and 2) provide a hierarchical structure, i.e. transfer (partial) trees of structures. This is motivated by the extremely small training corpus: more general (i.e. non-lexicalized) rules allow us to capture the structure of the training example independently of specific lexical items.

While the most specific rules are easily defined and understood, it is less clear how to define the most general rules that could be produced. Using similar reasoning as before, we can state that the most general rules that we could produce are ones whose component sequences contains exclusively constituent labels, so that every component must be filled by another rule. Clearly, not all rules in the grammar can be at this most general level. This definition is merely meant to emphasize once again that the rules are of varying degrees of generality. Compositionality Learning must determine what level of generality is appropriate for a given rule.

7.2.3 Space Defined by Rule Formalism

Before learning structural transfers, it is necessary to define a set of tags that describe categories of words or higher-level structures. The rule formalism developed for this work allows for two types of categories in addition to lexical items. The categories are as follows:

1. Non-terminals (NT): non-terminals are those labels that appear in the type of a rule. In our system, the non-terminals are the set of labels listed in Figure 4.2. Non-terminals are used in two rule parts: first, in the type definition of a rule (both for SL and TL, meaning X0 and Y0), and second in the component sequences for both languages. Non-terminals can be defined as any label that can be the type of a rule. They describe higher-level structures such as sentences (S), noun phrases (NP), or prepositional phrases (PP). Generally, they can be filled with more than one word. In other words, NTs in component sequences are filled by other rules.

2. Pre-terminals (PT): pre-terminals are part of speech labels. Pre-terminals can only be used in the component sequences of the rules, and not as X0 or Y0 types (except in the lexicon). Pre-terminals in component sequences can be filled by only one lexical entry, not by other grammar rules.

3. Terminals (LIT): terminals are lexicalized entries in the component sequences, and can be used on both the x- and the y-side. They can only be filled by the specified terminal itself.

These categories in combination with the definition of the rule formalism as well as the implementation of the transfer engine narrow the space of possible rules that can be learned. They enforce that there is a hierarchy of levels of abstraction, as specified by the categories.

The rule formalism furthermore implies a number of restrictions on the types of rules that the transfer engine can translate. Note that these restrictions are independent of any restrictions imposed by the learning algorithms on what rules could be learned from data. In fact, many rules could be learned, but could not be used for run-time translation, because they are meaningless in our system. This will become clearer further along in the discussion. The restrictions imposed by the rule formalism are subtle, but have wide-reaching implications. They are as follows:

NTs must not be aligned one-zero or zero-one.

This can be explained as follows: Suppose NTs could be aligned one-zero or zero-one. If it is one-zero aligned, then we arrive at a contradiction. Consider the following (abstract) rule that contains a one-zero alignment of an NT:

```
NT1::NT1 [PT1 NT2] -> [PT1]
((X1::Y1))
```

This rule may at first glance seem like a meaningful rule. Note, however, that it can never be applied. In order to resolve (fill) NT2 with actual words, it would need to apply another grammar rule with NT2 as its x-side type and no y-side type (as NT2 does not align to anything). The rule formalism enforces that each grammar rule must have an x-side *and* a y-side type. Hence, this rule can never resolve the NT2 and can thus never apply to produce a translation.

A similar argument holds for the zero-one alignment case. The only difference being that the grammar would need to contain a rule without a x-side type. The rule formalism does not allow this, hence a zero-one alignment is impossible.

PTs behave in a similar way to NTs:

PTs must not be aligned one-zero or zero-one.

Again, we first discuss the case of one-zero alignments. Consider the following (abstract) rule that contains a one-zero alignment of a PT:

```
PT1::PT1 [PT2 NT1] -> [NT1]
((X2::Y1))
```

In order for this rule to apply, there would need to be a lexicon entry with PT2 as its x-side top-node, but without a y-side top-node. This is not allowed in the transfer rule formalism, so that such a rule cannot exist in the lexicon, and the above rule can never apply. A zero-one alignment for a PT is equally impossible. It would require a lexicon entry with an empty x-side top-node, which is not allowed in the transfer rule formalism. Thus, a rule with a zero-one alignment for a PT can never apply. In practice, unaligned PTs result in undefined transfer engine behavior and should thus never occur in any rules.

Given these two restrictions, we can further conclude the following:

Any word in the bilingual training pair must participate in exactly one LIT, PT, or NT.

This principle follows from the fact that the training examples are used to infer transfer rules. In this process, each input word either participates in a consistent (NT), or it is generalized to the POS level (PT), or else it remains lexicalized (LIT). There are no circumstances under which the learning algorithm would have an input word participate in more than one component, or else the inferred rule would not reflect what was given in the training example, leading to undesired effects at run-time.

Although these three principles limit the space of meaningful rules, we have not yet developed a clear picture of what kinds of structures *are* possible in the given transfer rule formalism. In order to obtain a more complete picture, the space of possible rules can be expressed as a set of rewrites, as follows:

1. **X-side context-free rewrite rule.** The X0 node, i.e. the top-level node for the x-side, is transformed into the x-side component sequence.

2. **Transfer X0 to Y0.** The X0 node, i.e. the top-level node for the x-side, is transferred to the Y0 node, i.e. the top-level node for the y-side.

3. **Y-side context-free rewrite rule.** The Y0 node, i.e. the top-level node for the y-side, is transformed into the y-side component sequence.

4. **Transfer x-side component sequence to y-side component sequence.** The x-side component sequence is transferred to the y-side component sequence.

It is not obvious why these four rewrites do in fact capture the entire space of possible transfer rules under the rule formalism, so we will discuss why this is the case. In principle, the first two rewrites are exactly the language defined by a context-free grammar that transforms the top nodes into productions (or component sequences, as they are referred to in our formalism). The first rewrite captures what x-side component sequences can be produced from the X0 node, the second captures what y-side component sequences can be produced from the Y0 node.

The first and third rewrites would be enough to capture the entire space of possible transfers, were it not for the fact that the x-side and y-side are by no means independent. In fact, each transfer rule is tied to the other

language by the transfer of X0 to Y0 (the second rewrite), and the transfer of the x-side component sequences to the y-side component sequences (the fourth rewrite): these rewrites capture what structures can possibly occur together in one specific rule. For example, in the previous section we have proven that it is not possible for NTs or PTs to be aligned to the empty word. These kinds of restrictions can only be captured via these two rewrites between the x-side and the y-side.

We can now describe the four above rewrites in terms of the categories described above (NTs, PTs, LITs), as well as an additional empty word, ϵ.

- **X-side context-free rewrite rule.**
  - NT $\rightarrow$ (NT | PT | LIT)+
- **Transfer X0 to Y0.**
  - NT $\rightarrow$ NT
- **Y-side context-free rewrite rule.**
  - NT $\rightarrow$ (NT | PT | LIT)+
- **Transfer x-side component sequence to y-side component sequence.**
  - NT $\rightarrow$ NT+
  - PT $\rightarrow$ PT+
  - LIT $\rightarrow \epsilon$
  - $\epsilon \rightarrow$ LIT
    LITs are always considered unaligned by the transfer engine.

Any learning algorithm that we design must stay within the parameters of what the transfer formalism allows. Otherwise, we would get rules that would be meaningless to the transfer engine, as explained above. In the following settings, we will frame the learning algorithm within these constraints.

7.2.4 Space Defined by Learning Setting

The rule formalism imposes certain constraints on what rules can be handled. Similarly, the learning setting imposes a set of constraints. For convenience, we summarize below in one list what has been said in various places above.

The learning setting makes the following assumptions:

1. The presence of a TL parser.
2. The absence of an SL parser.
3. Word alignments are given for the training data.
4. We define the set of non-terminals, i.e. the types of rules that can be learned, as the set NT. In our system, the set of NTs consists of the types listed above in Figure 4.2: ADVP, ADJP, NP, PP, SBAR, and S, with subtypes. We discuss in chapter 4 the motivation for restricting the system to those types.

The assumption with the strongest implications is the absence of an SL parser. For this reason, we generally assume that a TL NT translates into the same NT on the SL side, as was discussed in chapter 6. In other words, since we only have a parser for the major language (TL), we assume that a given NT (such as NP) will translate into a the same NT (i.e. NP) on the y-side.

An TL NT is assumed to translate into the same NT in SL.

This assumption has wide-reaching implications, as it restricts the types of structures that can be learned. This simplifies learning, but it also limits the system.

Folding the above assumption into the four rewrites yields the following:

- **X-side context-free rewrite rule.**
  - NT $\rightarrow$ (NT | PT | LIT)+ (unchanged)
- **Transfer X0 to Y0.**
  - $NT_i \rightarrow NT_i$ (*same* type of NT)
- **Y-side context-free rewrite rule.**
  - NT $\rightarrow$ (NT | PT | LIT)+ (unchanged)
- **Transfer x-side component sequence to y-side component sequence.**
  - $NT_i \rightarrow NT_i$+ (*same* type of NT)

- PT → PT+ (unchanged)
- LIT → ϵ (unchanged)
- ϵ → LIT (unchanged)

Closer inspection of these transformations shows that the first and third rewrites do not pose any real restrictions on the rule learning algorithm. The only restriction is that at least one index must be produced, but this index can be filled by a LIT, PT, or NT. Transformation I, on the other hand, is now quite restrictive. Future work will address the softening of the assumption of direct transfer of NTs.

The second and fourth rewrites can be viewed as expressing cross-lingual learning. Let us call this 'xy learning'. Further, consider the categories NT, PT, and LIT in a hierarchy where NT is the highest-level and LIT is the lowest-level category. Finally, let a 'sequence' be defined as one or more instances of one category. Then a closer inspection of the second and fourth rewrites allows us to postulate the following axioms:

1. **A NT cannot map down.** A NT must map to another NT *of the same type*. It cannot map to a sequence of PTs or LITs, and it cannot map to the empty word ϵ.

2. **A PT cannot map up down.** A PT cannot map to a sequence of LITs, and it cannot map to the empty word ϵ. Corollary of axiom 1: **A PT cannot map up**. It cannot map to a sequence of NTs.

3. **A LIT can map to the empty word ϵ, and the empty word ϵ can map to a LIT**. Literals can only map to the empty word ϵ[1]. Corollary of axioms 1 and 2: **A LIT cannot map up.**

The last axiom may be the cause of confusion. While it happens frequently that one word in a language translates into more than one word in another language, it is undesirable to express such a transfer as the transfer for a specific word to a constituent. For example, the Hebrew possessive particle 'LI' generally translates into the English 'TO ME'. A learned rule would leave all three words lexicalized, rather than proposing a transfer of 'LI' to a PP. The PP could not be filled by any rule, because it would not have a corresponding x-side.

Furthermore, the above discussion leads us to the following additional axiom. Although this was discussed above, it should be repeated here as an axioms, as all learned rules must abide by these axioms:

[1]Remember that literals are always considered unaligned by the transfer engine.

4. **Any word must participate in exactly *one* LIT, PT, or NT**. This follows from the fact that the transfer engine matches against the input linearly. See section 7.2.3 for a more detailed discussion.

These axioms stem from a combination of 1) the transfer rule formalism and transfer mechanism and 2) the restrictions imposed in the absence of the transfer engine. In fact, we could easily design a learning algorithm that would propose rules that violate these axioms, but the system will never be able to apply them properly. Therefore, any learning algorithm must either be designed to never violate these axioms, or else must filter out all learned rules that violate one or more of the axioms, because such rules will merely slow the system without any added benefit.

The question then becomes how to define a learning mechanism that will learn reasonable structures while at the same time 'playing' within the rules of the game, i.e. not violating any of the axioms. In the sections below it will become clear how our learning algorithms ensure that all learned rules abide by the axioms.

7.3 The Basic Compositionality Algorithm

In order to exploit the inherent structure in the training examples, and in order to scale better to unseen examples, the system learns compositional rules. The goal of this module is to learn the context-free backbone of the transfer grammar. This is achieved by traversing the TL parse from the top-down, and introducing a compositional element for a subnode wherever appropriate. Two basic settings are possible for Compositionality learning: assuming Maximum Compositionality, or using the transfer engine and previously learned rules to decide whether compositional elements should be introduced. We will discuss these settings in the following two sections. This section lays out the algorithm independently of the Maximum Compositionality parameter.

In the introduction to this chapter, we presented a flat seed rule and several possible generalizations of this rule that were compositional in that their component sequences contained constituent labels. How can such compositional rules learned? This can be accomplished by traversing the TL parse from the top down. For each subnode, the algorithm decides whether or not to introduce a compositional element. This decision depends first on whether Maximum Compositionality is assumed or not. It further depends on subtle issues such as whether the TL component aligns to a contiguous set

of SL words, or whether the aligned SL words align to other TL components. These advanced issues will be described below.

The algorithm will first try to introduce a compositional element for each of the direct children of the root. The root itself is not considered for Compositionality, because we do not want to introduce completely recursive rules of the form

```
S::S [S] -> [S]
(
(X1::Y1)
)
```

Such rules do not add any new information to the grammar, and will make the run-time infinite by applying repeatedly.

For each direct child of the node, a compositional element can be introduced. If this happens, no further traversal of the subtree rooted at the child takes place. Otherwise, the Compositionality algorithm descends further into the tree, with a base case of leaving the seed rule non-compositional.

What exactly does it mean to introduce a compositional element? The label of the compositional subroot replaces a substring of the component sequences (in the appropriate place) of the rule. As was discussed above, we always require that the label of a compositional element is the same both for the SL and TL. This is necessary so that other rules can 'plug into' the now compositional rule. For example, if a compositional element is of the type NP, then this NP can be resolved at run-time using another NP::NP rule, as in a standard context-free grammar.

Since the Compositionality algorithm traverses the parse from the top-down, it prefers the most far-reaching compositional elements. For instance, if the S 'THEY RETURNED' qualifies for compositionality, then the lower-level NP 'THEY' is no longer even considered for compositionality. This again illustrates the fact that Compositionality aims at learning rules that are as general as possible while still supported by the training data.

In figure 7.1, we present the basic Compositionality algorithm in pseudocode. As will be explained in greater detail below, whether compositionality is valid for a given subroot depends in part on whether Maximum Compositionality is assumed or not. If Maximum Compositionality is assumed, compositionality is always assumed valid; if not, then compositionality is only valid if a previously learned rule can translate the TL sequence that the subroot spans into the corresponding SL sequence.

In the following, we will discuss two approaches to Compositionality Learning. In addition to the running examples throughout the chapter, we

will present a number of learned rules in section 7.8.1. There, we will compare non-compositional rules to compositional rules, as well as compositional rules that were learned using one of the two approaches as described in the following sections.

The complexity of this algorithm depends to a large extent on the specific approach that is taken. When Maximum Compositionality is not assumed, Compositionality Learning is much less efficient than when Maximum Compositionality is assumed. In the following two sections, we will discuss the complexity of the algorithms. Complexity is however only an issue under the first approach.

7.4 Approach I: Learning Without Maximum Compositionality

In the learning setting where Maximum Compositionality is no assumed, the rule learner uses previously learned rules to determine whether or not a compositional element should be introduced. Note that under this paradigm, it is very important what order the rules are learned in. Sections 7.4.1 and 7.4.2 discuss how the appropriate order can be ensured.

How can the rule learner use previously learned rules to determine whether a compositional element should be introduced? First, each of the nodes in the parse is used to 1) extract the TL substring ($w_{1,TL} \ldots w_{k,TL}$ in the pseudocode above) covered by this node 2) using the alignments, create a SL substring corresponding to the TL chunk ($w_{1,SL} \ldots w_{l,SL}$ above), 3) the type of the node ($label_{Subroot}$).

The learning algorithm then uses all rules of type $label_{Subroot}$ to translate the TL chunk that was extracted from the training example. Only rules of this type can serve as the top-node in this translation. The rules can however combine with rules of any other type to form a translation of $w_{1,TL} \ldots w_{k,TL}$. If any of the possible translations is the correct translation $w_{1,SL} \ldots w_{l,SL}$, then a compositional element can be introduced if $w_{1,SL} \ldots w_{l,SL}$ is contiguous. To return to the previous example, consider the node and subtree

```
(<S> (<NP> (PRO they-4))
     (<VP> (V returned-5))))
```

The Compositionality algorithm extracts the TL chunk 'THEY RETURNED' and, using the alignments, the SL chunk 'HM $BW'. Then the transfer engine uses all previously learned rules of type S to translate 'THEY

Basic Compositionality Algorithm

```
IntroduceCompositionalElement(Root), where Root is
the root of the TL parse

IntroduceCompositionalElement(Subroot)

    Get parse label label_Subroot of Subroot
    Extract w_a,TL ... w_b,TL covered by Subroot and
    their corresponding entries comp_a,TL ... comp_b,TL
    in the TL component sequence
    Extract w_c,SL ... w_d,SL aligned to w_a,TL ... w_b,TL and
    their corresponding entries comp_c,SL ... comp_d,SL
    in the SL component sequence
    If w_c,SL ... w_d,SL contiguous and w_c,SL ... w_d,SL ≠ ϵ
        If compositionality valid for Subroot
            Replace comp_a,TL ... comp_b,TL by label_Subroot
            in the TL component sequence
            Replace comp_c,SL ... comp_d,SL by label_Subroot
            in the SL component sequence
            Adjust the alignments in the rule
        Else for each child of Subroot
        Child_i,Subroot, a ≤ i ≤ b
            IntroduceCompositionalElement(Child_i,Subroot)
```

Figure 7.1: Pseudocode for Basic Compositionality. Whether compositionality is valid for a subroot depends on the algorithm chosen, as described in 7.4 and 7.5.

RETURNED'. If one of the possible translations is the expected 'HM $BW', a compositional element is introduced with the label 'S' both for TL and SL. Finding a correct translation for the TL chunk means that we have previously learned a rule of type S which, possibly in combination with other previously learned rules, can translate this chunk. Two exceptions hold: first, the SL chunk must be contiguous, and second, a compositional element is not introduced when the TL constituent is not aligned to any word in SL. In this case, it is safer to simply leave this sequence lexicalized[2].

What are the previously learned rules? Rules are learned in several iterations according to their co-embedding score and type, as will be described below (sections 7.4.1 and 7.4.2). As before, Compositionality begins from the top of the parse and therefore introduces compositional elements that are as big as possible.

The advantage of this approach is that we only introduce compositional elements that are warranted by the previously learned rules and thus is directly 'endorsed' by the training data. The disadvantage is that there is a strong reliance on structural variety in the training data: if a certain structure was not encountered in the training data by itself, it cannot be recognized as a compositional element in the context of a different rule. Further, compositionality can also suffer if the training data contains noise. We can expect that this will almost always be the case to a certain extent.

The generic Compositionality algorithm listed above is modified slightly in the pseudocode in Figure 7.2 to reflect that Maximum Compositionality is not assumed.

Throughout this document, we have noted that the algorithms laid out here are designed to extract the maximum amount of information from a minimum amount of training data. For this reason, training complexity is never a big issue in practice. However, it should be mentioned that out of all the algorithms presented in this thesis, the Compositionality algorithm without the assumption of Maximum Compositionality is the most expensive in terms of time complexity, and could cause problems if training data were increased by a large factor. In practice, the complexity of the run-time system far outweighs the training time for any of the algorithms.

The worst-case complexity of this algorithm can be analyzed as follows. Suppose that the number of training examples is n. Then in the worst case,

[2]It can be argued that training examples with unaligned constituents will always result in bad learned rules. This is not necessarily the case. For example the rule ADJP::ADJP ["BN" NP] → [NP 'OLD'] ((X2::Y1)) contains the unaligned constituent 'OLD'. The translator did not align the two remaining literals in the rule because 'BN' literally means 'SON'. Nevertheless, the learned rule is valid and useful.

Compositionality Algorithm Without Assuming Maximum Compositionality

```
IntroduceCompositionalElement(Root), where Root is
the root of the TL parse

IntroduceCompositionalElement(Subroot)

    Get parse label label_Subroot of Subroot
    Extract w_{a,TL} ... w_{b,TL} covered by Subroot and
    their corresponding entries comp_{a,TL} ... comp_{b,TL}
    in the TL component sequence
    Extract w_{c,SL} ... w_{d,SL} aligned to w_{a,TL} ... w_{b,TL} and
    their corresponding entries comp_{c,SL} ... comp_{d,SL}
    in the SL component sequence
    If w_{c,SL} ... w_{d,SL} contiguous and w_{c,SL} ... w_{d,SL} ≠ ϵ
        If there exists a previously learned rule
        of type label_Subroot that can translate
        w_{a,TL} ... w_{b,TL} into w_{c,SL} ... w_{d,SL}
            Replace comp_{a,TL} ... comp_{b,TL} by label_Subroot
            in the TL component sequence
            Replace comp_{c,SL} ... comp_{d,SL} by label_Subroot
            in the SL component sequence
            Adjust the alignments in the rule
        Else for each child of Subroot
        Child_{i,Subroot}, a ≤ i ≤ b
            IntroduceCompositionalElement(Child_{i,Subroot})
```

Figure 7.2: Pseudocode for Compositionality without the Maximum Compositionality Assumption.

each training example will result in a unique seed rule. For each seed rule, the Compositionality algorithm is run, which involves checking for the existence of lower-level rules that can translate chunks of the training example. Denote the maximum tree depth of any tree in the training data by d, and denote the maximum number of non-root, non-terminal nodes in any parse tree in the training data by k. Then the Compositionality algorithm without maximum compositionality is bound above by $n * ((k * (n-1)) * ((d-1) * (n-1)))$. For each training example, the Compositionality algorithm must check up to $n-1$ existing rules. This, however, is not enough, because the existing rules may have to combine with other existing rules (including themselves) in order to translate the chunk in question. The number of such combinations is however bounded above by the depth of the tree[3], because the chunk in question can only require as many rules as the depth of its parse tree. In big-O notation, the complexity of this algorithm can thus be summarized as $O(n^3)$. This could create problems if the training data was very large. As was said above, in practice, complexity was not an issue at training time. Should it become a problem, the rule learner could either be run using the Maximum Compositionality assumption.

7.4.1 Iterative Type Learning

It was mentioned above that we concentrate our learning efforts on a set of constituents that are of high interest to English and can furthermore be assumed to transfer into another language with minimal structural divergence. We concentrate on the following types (with some subtypes). For examples of each of these types and a discussion of why these types were chosen, see Figure 4.2. The types are based on the nodes marked in the Penn Treebank (Marcus et al., 1995).

1. ADVP
2. WHADVP
3. ADJP
4. WHADJP
5. NP
6. WHNP
7. PP
8. WHPP
9. SBAR
10. SBARQ

[3]Unless there are circular rules such as NP [NP] → [NP], which are pruned out, as will be described in section 7.7.

11. S
12. SQ
13. SINV

This list can easily be changed for a different language pair. However, it is assumed that it will stay relatively constant whenever the TL is English.

In this table, we list the types of interest in a particular order: this is the order in which these types are learned. When Maximum Compositionality is not assumed, it makes sense to learn 'simpler' constituents, such as ADJPs first, and then to move on to more complex constituents that are likely to embed the simpler ones. For example, it is by far more likely that an NP will embed an ADJP than the other way around. By enforcing this order, the rules are in fact learned 'bottom up', i.e. from simplest to most complex.

When the Compositionality algorithm (without assuming Maximum Compositionality) is run on an NP that embeds an ADJP, this ADJP can only be recognized as a compositional element embedded within the NP if a rule has already been learned for the ADJP. The order described above maximizes the probability that types that embed other constituents are learned *after* the embedded constituents are learned.

7.4.2 Co-Embedding Resolution

While iterative type learning is a step in the right direction, it cannot resolve all embedding conflicts. We mentioned above that it is more likely that an NP embeds and ADJP than the other way around. This is true, but the opposite can still happen, as in the example 'PROUD OF THE CHILDREN'. Under iterative type learning, ADJPs are learned *before* NPs and PPs are learned, and thus neither 'OF THE CHILDREN' nor 'THE CHILDREN' can be recognized as compositional elements and inserted into the ADJP as such.

In order to overcome this difficulty, we tag each training example with a so-called *co-embedding score*. The co-embedding score is the depth of the TL parse tree.[4] Learning is then run from the smallest co-embedding score to the highest co-embedding score. In each iteration, i.e. for each co-embedding score, Seed Generation and Compositionality are performed for each example that is of the given co-embedding score. Within one iteration, training examples are considered in the order that is imposed by iterative

[4]In previous versions, the co-embedding score was the number of embedded constituents of the same type as the root label. It was found that this does not indeed solve all co-embedding problems, as for instance in the example 'PROUD OF THE CHILDREN'

Co-embedding Resolution and Iterative Type Learning

```
Find highest co-embedding score in training
data, max_coembedding
Find the number of types to learn, n_types
For (1 ≤ i ≤ max_coembedding)
    For (1 ≤ j ≤ n_types)
        For all training examples with co-embedding
        score i and of type type_j
            Perform Seed Generation
            Perform Compositionality Learning
```

Figure 7.3: Pseudocode for Co-Embedding Resolution and Iterative Type Learning.

type learning. In other words, we first learn rules for all training examples of co-embedding score 1, in the order ADVP, ADJP, NP, ... S. This is followed by another pass, in which we learn rules for all training examples of co-embedding score 2, again in the order ADVP, ADJP, NP, ... S. The process repeats up to the highest co-embedding score found in the training data. Since learning is efficient both in terms of memory and time usage, running several iterations does not pose a problem, and Compositionality learning is ensured to always be able to draw on all potentially relevant rules for a given context.

Co-embedding resolution and iterative type learning together result in two big loops around Seed Generation and Compositionality Learning, as can be summarized in pseudocode as in Figure 7.3.

7.5 Approach II: Learning With Maximum Compositionality

An alternative to the algorithm that uses previously learned rules is the assumption of Maximum Compositionality. Under this paradigm, the rule learner does not check whether there exists a previously learned rule that can translate the TL chunk into the SL chunk. Instead, we make the assumption that under 'ideal' circumstances, this will be the case. If the training data were noise-free and contained all possible compositional elements that might be needed by higher-level rules, then the assumption of Maximum Compo-

sitionality would hold. This means that every direct child of the parse root is eligible for compositionality, and generally, a compositional element is introduced. As before, a compositional element is not introduced if the SL chunk is not contiguous, or if the TL constituent is not aligned to any word in SL, and the unaligned sequence is left lexicalized.

More formally, the Compositionality algorithm with the assumption of Maximum Compositionality can be given as follows:

For every direct child of the parse root, introduce a compositional element if

1. ***the TL words covered by this direct child*** $w_{TL,k} \dots w_{TL,m}$ ***align to a coherent chunk of SL words*** $w_{SL,k} \dots w_{SL,m}, k < m$***, i.e. no words*** $w_{SL,l}, k < l < m$ ***are aligned to a word outside*** $w_{TL,k} \dots w_{TL,m}$***.***
2. ***the aligned chunk is not necessarily contiguous, but can only be interrupted by unaligned SL words.***

Let us once more return to the previous example:

```
TL: a year later they returned
SL: $NH MAWXR IWTR HM $BW
Alignment: ((2,1),(3,2),(3,3),(4,4),(5,5))
Type: S
CoEmbeddingScore: 4
C-Structure:
 (<S> (<ADVP> (<NP> (DET a-1)(N year-2))
              (ADV later-3))
      (<S> (<NP> (PRO they-4))
           (<VP> (V returned-5))))
```

The immediate children of the parse root are and ADVP and a S. The ADVP covers the TL chunk 'A YEAR LATER' with the corresponding SL chunk $NH MAWXR IWTR'. Since '$NH MAWXR IWTR' is *contiguous*, and the TL chunk is not unaligned, a compositional element can be introduced.

Similarly, the other direct child of the parse root, S, covers the TL chunk 'THEY RETURNED', with the corresponding translation 'HM $BW'. Again, the TL chunk is not unaligned, and the corresponding SL chunk is contiguous, so compositionality is valid.

The two compositional steps lead to the following maximally compositional rule:

```
;;SL: $NH MAWXR IWTR HM $BW
;;TL: A YEAR LATER THEY RETURNED
S::S [ADVP S] -> [ADVP S]
(
(X1::Y1)
(X2::Y2)
)
```

The advantage of the Maximum Compositionality approach is that the algorithm is no longer limited by what is observed in the training data, i.e. all compositional structures were previously observed. The potential fallacy is overgeneralization: under this paradigm, the Compositionality algorithm aggressively 'imposes' the TL structure onto the SL sentence (while allowing for reordering), which in some cases is incorrect. We will give a more detailed discussion of the advantages and disadvantages of Maximum Compositionality in sections 7.8.2 and 9.4 below.

Under the Maximum Compositionality assumption, the generic Compositionality algorithm simplifies to the pseudocode in Figure 7.4.

In the first approach, i.e. no assumption of Maximum Compositionality, we have argued that training time complexity could be an issue with large amounts of training data, as the algorithm scales in $O(n^3)$ time. One way to overcome this problem is the second approach, i.e. the assumption of Maximum Compositionality. Under this paradigm, the algorithm scales linearly: the number of seed rules is bounded above by the number of training examples, n. For each seed rule, there will be a constant number of non-root, non-terminal nodes in the parse tree. In the worst case (i.e. if compositionality cannot be introduced because of complex alignments), all non-root, non-terminal nodes may be explored by the Compositionality algorithm. However, no *other* rules must be examined. If you again denote the maximum number of non-root, non-terminal nodes in any parse tree in the training data by k, then the complexity of the Compositionality algorithm with the assumption of Maximum Compositionality is bounded above by $n * k$, which is $O(n)$.

7.6 Advanced Structural Learning

In this section, we describe several algorithms to deal with advanced cases of higher structural complexity.

This chapter has so far focused on cases where constituents translate in constituents of the same type. If this can safely be inferred, we propose a

Compositionality Algorithm With Assuming Maximum Compositionality

```
IntroduceCompositionalElement(Root), where Root is
the Root of the TL parse

IntroduceCompositionalElement(Subroot)

    get parse label label_Subroot of Subroot
    extract w_a,TL ... w_b,TL covered by Subroot and
    their corresponding entries comp_a,TL ... comp_b,TL
    in the TL component sequence
    extract w_c,SL ... w_d,SL aligned to w_a,TL ... w_b,TL and
    their corresponding entries comp_c,SL ... comp_d,SL
    in the SL component sequence
    if w_c,SL ... w_d,SL contiguous and w_c,SL ... w_d,SL ≠ ϵ
        If w_c,SL ... w_d,SL is coherent, i.e. no word
        ∈ w_c,SL ... w_d,SL aligns to a word outside of
        w_a,TL ... w_b,TL
            Replace comp_a,TL ... comp_b,TL by label_Subroot
            in the TL component sequence
            Replace comp_c,SL ... comp_d,SL by label_Subroot
            in the SL component sequence
            Adjust the alignments in the rule
        else for each child of Subroot
        Child_i,Subroot, a ≤ i ≤ b
            IntroduceCompositionalElement(Child_i,Subroot)
```

Figure 7.4: Pseudocode for Compositionality with the Maximum Compositionality Assumption. 'If $w_{c,SL} \dots w_{d,SL}$ is coherent' indicates the difference between Maximum Compositionality and the algorithm without Maximum Compositionality.

generalization to the constituent level. With this approach, we can capture the transfer of constituents, and we can capture in particular reorderings of constituents. This approach is supplemented by instances where we do *not* generalize to the constituent level: we either leave words lexicalized, or else we only generalize to the POS level. Such cases of structural differences are common when translating from one language into another. Different approaches have been proposed to account for such phenomena. Dorr et al. (Dorr et al., 2002) classify such differences in structural make-up into several categories of *divergences*: examples are categorial variation (where a word of a given POS translates into a word of a different POS), head swapping (where two phrases which are translations of each other have heads that are not translations of each other), etc. Others, such as Levin and Nirenburg (Levin & Nirenburg, 1994) take a different approach by accounting for the differences with a theory of *constructions*. A construction is a way in which a language expresses a certain concept. In a different language, the same concept may be expressed with a completely different syntactic make-up. The theories of constructions and divergences are not contradictory. They merely explain the same phenomena from different perspectives. The divergence approach captures the surface form of the expressions, whereas the construction approach focuses more on the concept that is being expressed.

It is not the purpose of this thesis to provide a full account of translation divergences or constructions. The phenomenon of translation mismatches, as a general concept, however plays a role in our work: in this chapter, we propose several methods to handle such translation mismatches. We present a method to handling noun compounds that translate into single nouns, a method for learning lexical entries for words that translate into a constituent of a certain composition, etc. Not all translation mismatches are handled here. However, this section addresses the issue of how such mismatches can be handled in principle. In the future work section (cf. section 10.3), we will discuss extensions to the algorithms described in this section.

All of the phenomena described below are rare, so that they do not impact the automatic scores. We felt, however, that it was important to develop algorithms that are able to deal with these cases. While the algorithms described below are thus mainly of theoretical interest, they prove that our system is able to handle complex cases. Furthermore, future research into these and other complex cases is facilitated, because we show below that it is possible to devise effective algorithms for such phenomena.

7.6.1 Pro-Drop

Pro-drop is a phenomenon common to many languages in which a personal pronoun can be dropped if it is implicit in the form of the main verb. As pro-drop is a common phenomenon across languages, we chose to handle it as a special case. Hebrew exhibits pro-drop under certain circumstances. For instance, the first person pronouns often drop in past tense sentences.

How pro-drop is handled in our system depends in part on alignment choices that the translator makes. If the translator only aligns the verbs to each other, the learning algorithm handles pro-drop naturally: the pronoun remains lexicalized in the rule.

If the translator however aligns both the English pronoun and the English verb to the Hebrew verb, then the basic learning system would leave all three words lexicalized. In this case, our approach to pro-drop consists of

- Recognizing an instance of pro-drop,
- Breaking the link between the pronoun and the verb, and
- Generalizing the now one-one aligned verbs to the POS level

In essence, this handling of pro-drop creates the rule that would have been learned had the translator aligned only the verbs and left the pronoun unaligned.

Pro-drop is handled before Compositionality Learning takes place, so that Compositionality can already take advantage of the pro-drop generalized seed rule. After a seed rule is produced, the learning algorithm checks whether it contains an example of pro-drop, i.e. an alignment of the SL main verb to both a pronoun and a verb on the TL side. Note that it is not immediate how the learner knows what (lexicalized) word on the SL is the main verb, especially because this word is aligned to two TL words of different POS. The SL word POS is obtained with the SL morphology module: if one of the possible POS for this word is a verb, and if the aligned TL words are a pronoun and a verb, then it is assumed that the training example exhibits pro-drop.

Once pro-drop is detected, the algorithm breaks the 'weaker' alignment link between the pronoun and the verb, and retains the alignment between the two verbs. This has the consequence that the TL pronoun remains lexicalized as before. One area for future work is to generalize this word to the POS level if it is found that appropriate constraints can be introduced. A rule that inserts a pronoun without constraints is undesirable, because all

the TL pronouns would be proposed for this slot at run-time. However, if the appropriate constraints that enforce agreement in number and person can be learned for the given rule, then generalization of the pronoun to POS is appropriate. This step would have to take place during Constraint Learning.

Removing the link between the TL pronoun and the SL verb causes the verbs to now be aligned one-one. This means that they can safely be generalized to the POS level.

An example of pro-drop can be found in the following training example:

```
TL: I come
SL: ABWA
Alignment: ((1,1),(2,1))
Type: S
CoEmbeddingScore: 3
C-Structure:(<S> (<NP> (PRO I-1))(<VP> (V come-2)))
```

Seed generation produces a seed rule that leaves both verbs as well as the pronoun lexicalized:

```
;;SL: ABWA
;;TL: I COME
;;Alignment:((1,1),(2,1))
;;CStructure:(<S> (<NP> (PRO I-1))(<VP> (V come-2)))
S::S ["ABWA"] -> ["I" "COME"]
(
;(X1::Y1)
;(X1::Y2)
)
```

The pro-drop handler breaks the link between the pronoun and the verb (as reflected in the alignments), and generalizes the verbs to the POS level:

```
;;SL: ABWA
;;TL: I COME
;;Alignment:((1,1),(2,1))
;;CStructure:(<S> (<NP> (PRO I-1))(<VP> (V come-2)))
S::S [V] -> ["I" V]
(
(X1::Y1)
)
```

Treatment of Pro-Drop

For all instances of one-two alignments, i.e. instances where a word w_{SL} is aligned to $w_{1,TL}, w_{2,TL}$

    Extract the entries in the component sequences $comp_{SL}$,$comp_{1,TL}$, $comp_{2,TL}$ corresponding to w_{SL},$w_{1,TL}$,$w_{2,TL}$, respectively
    If ((w_{SL} is aligned to $w_{1,TL}, w_{2,TL}$) && ($comp_{w_{SL}} = V$) && ($comp_{w_{1,TL}} = PRO$) && ($comp_{w_{2,TL}} = V$))
        Remove alignment between w_{SL} and $w_{1,TL}$
        Replace $comp_{TL}$ and $comp_{2,SL}$ in the rule with label V

Figure 7.5: Pseudocode for treatment of pro-drop.

Such a rule will allow the translator to now propose correct translations for instances of pro-drop. For example, the Hebrew 'AXLTI' (eat.past.1.sg) can now be translated into 'I ATE', not simply into 'ATE'.

This rule will not always perform well in practice. Ideally, Unification Constraint Learning will introduce the appropriate constraints in order to make this rule less general. In practice, the TL language model in the decoder is often enough to filter out bad applications of this rule.

Figure 7.5 lists the treatment of pro-drop in pseudocode.

7.6.2 Compounds and other One-Many Alignments

One-many alignments are frequent in our training data. For example, many English nouns translate into a compound of two nouns in Hebrew, such as 'HOSPITAL' → 'BT' 'XWLIM' (house sick.pl.masc). Denote the TL word by w_{TL}, and denote the sequence of SL words by $w_{1,SL} \dots w_{n,SL}$. We then handle one-many alignments as follows: If the aligned word sequence in SL is contiguous, we check whether there exists an entry in the dictionary that equates the SL sequence $w_{1,SL} \dots w_{n,SL}$ with the TL word w_{TL}. If there is such an entry, then we replace the word sequence $w_{1,SL} \dots w_{n,SL}$ with the SL type of the dictionary entry, and replace w_{TL} with the TL type of the dictionary entry.

For example, consider the rule

```
;;SL: B BT XWLIM
;;TL: IN A HOSPITAL
;;Alignment:((1,1),(3,2),(3,3))
;;CStructure:(<PP> (PREP in-1) (<NP> (DET a-2) (N hospital-3)))
S::S [PREP "BT" "XWLIM"] -> [PREP "A" "HOSPITAL"]
(
(X1::Y1)
;(X2::Y3)
;(X3::Y3)
)
```

if there exists a dictionary entry for 'HOSPITAL' - 'BT' 'XWLIM', then with compound handling, the rule becomes:

```
;;SL: B BT XWLIM
;;TL: in a hospital
;;Alignment:((1,1),(3,2),(3,3))
;;CStructure:(<PP> (PREP in-1) (<NP> (DET a-2) (N hospital-3)))
S::S [PREP N] -> [PREP "A" N]
(
(X1::Y1)
(X2::Y3)
)
```

This rule is clearly more general. It can now apply to any N::N entry in the dictionary, such as 'BT'→'HOUSE', which could not have been handled by this rule before applying the compound algorithm.

In Figure 7.6, the algorithm is given in pseudocode.

This approach handles most one-many alignments that we encounter for Hebrew→English translation. There are few examples that do not fall into this category, and they cannot currently be handled in our system, so that they are left lexicalized. Consider the following example:

```
;;SL: &WRK H DIN $ RAITI
;;TL: THE LAWYER WHO I SAW
NP::NP ["&WRK" DET "DIN" SUBORD "RAH"] ->
                                [DET "LAWYER" SUBORD "I" "SAW"]
(
;(X1::Y2)
(X2::Y1)
```

Treatment of Compounds

For all instances of many-one alignments, i.e. instances where a sequence $w_{a,SL} \dots w_{b,SL}$ is aligned to w_{TL}

- Extract the entries in the component sequences $comp_{a,SL}$ ... $comp_{b,SL}$ and $comp_{TL}$ corresponding to $w_{a,SL} \dots w_{b,SL}$ and w_{TL}, respectively
- If the sequence $w_{a,SL} \dots w_{b,SL}$ is contiguous
  - Check the translation ‘‘$w_{a,SL} \dots w_{b,SL}$’’→‘‘w_{TL}’’ in the dictionary. If it exists with types $type_{SL} :: type_{TL}$
    - Replace $comp_{a,SL}$ ... $comp_{b,SL}$ in the SL component sequence with $type_{SL}$ and replace $comp_{w_{TL}}$ in the TL component sequence with $type_{TL}$.
  - Else leave the words $w_{1,SL} \dots w_{n,SL}$ as well as w_{TL} lexicalized in the component sequences
- Else leave the words $w_{1,SL} \dots w_{n,SL}$ as well as w_{TL} lexicalized in the component sequences

Figure 7.6: Pseudocode for treatment of compounds.

```
;(X3::Y2)
(X4::Y3)
)
```

The current version of the transfer engine processes words linearly and cannot recognize and translate discontinuous compounds. In order to overcome this difficulty without a change in the transfer engine, we would need to pre-process the data and reorder the words before they are passed to the transfer engine: '&WRK' 'H' 'DIN', if encountered at run-time, would then be passed into the transfer engine as 'H' '&WRK' 'DIN'. This is not a trivial task, in particular for Hebrew, because Hebrew morphology is highly ambiguous, and many words can be interpreted as a sequence of the parts of speech N 'H' N. As a robust alternative, the input sentence could be passed to the transfer engine in both original and reordered form, and the resulting lattices could be probabilistically merged. While this is an interesting area for future investigation, it is outside the scope of this thesis, and such cases remain lexicalized for all learned grammars that are reported in this document.

7.6.3 Lexicon Enhancement

For some language pairs, the translation of certain words of one part of speech consistently yields an expression involving a different part of speech. For example in Hebrew, adverbs are formed by a combination of the preposition 'B' and a noun. For example, 'HAPPILY' translates into 'B SMXH', literally 'IN HAPPINESS'. In this section, we aim at automatically discovering such cases, and handling them in the rule learner. We will exemplify the approach with Hebrew adverbs simply because it is the most prominent example of a POS shift between Hebrew and English. The approach is however more general and can be used to automatically detect and deal with POS shifts.

The basic rules learned for cases such as this one would contain the adverb, as well as the preposition and the noun in Hebrew, in a lexicalized form. It would be much better if the rule learner could determine that an English adverb always (or at least in most cases) translates into the preposition 'B' and a noun.

We attack this problem in a two-step process, one involving the lexicon and one involving the learned rules. The first step is the lexicon enhancement step. We scan through the training data looking for instances of one-many alignments. Whenever we encounter that a specific POS transfers

consistently into a specific sequence of POS labels, we initiate a lexicon enhancement step. This is especially applicable in the Hebrew adverbs case, because the English adverb varies together with the Hebrew noun, whereas the Hebrew preposition is constant.

While will now explain this process for the example of ADV → 'B' N, we want to stress however that this process could be applied in the more general case to other language pairs and other POS sequences. Below, we describe the results on running the automatic enhancement algorithm on the training data.

The algorithm proceeds as outlined in pseudocode in Figure 7.7. Denote the varying TL word $v_{i,TL}$ and the varying SL word $v_{i,SL}$, and denote the constant SL word c_{SL}. The algorithm is seeded with all instances of a one-two alignments where one word is constant and the other words are of a specific POS. It then finds a derivational morphology rule which allows it to propose additional lexical rules.

As an additional precautionary step, the newly produced TL word is be checked against a monolingual TL corpus or against a monolingual TL dictionary, and the new lexical rule is only retained if the new word appear in the dictionary or corpus. Note that TL is assumed to be a major language, so it is fair to assume that such a corpus or dictionary will exist. In the case of TL being English, we can check whether the newly created TL word appears in the British National Corpus (Leech, 1992),(Leech et al., 1994). Using the BNC, we can also verify that the new word is of the desired part of speech.

For illustration we now demonstrate the algorithm with the above example 'HAPPILY' → 'B' 'SMXH'.

1. Get all translations $w_{1,TL} \dots w_{n,TL}$ of the "SMXH" from the lexicon. $w_{1,TL}$ = "JOY" and $w_{2,TL}$ = "HAPPINESS".

2. Using minimal edit distance, find the translation $w_{closest,TL}$ that is closest to the varying "HAPPILY". $w_{closest,TL}$ = "HAPPINESS".

3. Learn a replacement rule (a derivational morphology rule) to modify "HAPPINESS" into "HAPPILY". The replacement rule in this case is "INESS" → "ILY".

4. Apply the learned rule to all N::N entries in the dictionary and introduce new entries of type ADV::ADV.

The list of new entries can either be manually inspected by a native bilingual speaker, or can simply be introduced into the lexicon. Since the

Lexicon Enhancement Using Training Data

For all sets of tuples of the form C=$\{\langle c_{SL}, v_{i,SL}, v_{i,TL}\rangle$, $1 \leq i \leq |C|$ | $v_{i,TL}$ aligns to c_{SL} and $v_{i,SL}$ and c_{SL} is constant $\forall i\}$

For each varying word $v_{i,SL}$

Get all translations $w_{1,TL} \ldots w_{n,TL}$ of $v_{i,SL}$ from the lexicon.

Using minimal edit distance, find the one translation $w_{closest,TL}$, $w_{closest,TL} \in w_{1,TL} \ldots w_{n,TL}$ that is closest to the varying TL word $v_{i,TL}$.

Learn a replacement rule (a derivational morphology rule, e.g., Adj:‘‘’’→‘‘ly’’) to modify $w_{i,TL}$ into $v_{i,TL}$.

For all words $w_{1,SL} \ldots w_{m,SL}$ in the lexicon that are of the same POS as $v_{i,SL}$, get all translations $t_{1,TL} \ldots t_{m,TL}$.

For all words $t_{1,TL} \ldots t_{m,TL}$, apply the replacement rule to obtain modified words $t'_{i,TL}$. Note that the replacement rule will not apply to all words $t_{1,TL} \ldots t_{m,TL}$.

If (($t'_{i,TL}$ appears in a monolingual TL dictionary or corpus) && (One of the possible POSs for $t'_{i,TL}$ according to the monolingual TL dictionary or corpus is $v_{i,TL}$'s POS))

For each modified word $t'_{i,TL}$, introduce a new lexical item of the form ‘‘c_{SL} $w_{i,SL}$’’ → ‘‘$t'_{i,TL}$’’. The type of the entry will be the same POS as the type of $v_{i,TL}$.

Figure 7.7: Pseudocode for lexicon enhancement.

algorithm is completely automatic, some spurious rules are introduced. We have a mechanism to prevent the introduction of invalid English words (see below). Also, if the Hebrew sequence is invalid, the lexical rule will never apply, because the sequence will not be encountered at runtime. However, as in any automatic process, we cannot guarantee that all of the newly created entries are actual correct translations.

Returning to rule learning, we can now apply a similar approach to the one described for compounds above: if we encounter a seed rule with three lexicalized words that align to each other, and if there is a lexicon entry that can translate the two words in SL into the corresponding TL word, then we replace the sequence of SL words as well as the single TL word with the POS that is found in the dictionary. For this case, rather than leaving the words lexicalized, we will now replace them with ADV on both sides, and align the indices one-one. In this way, any ADV::ADV lexicon entry can fill these indices (i.e. components), and we have successfully generalized from the lexical level as appropriate.

The lexicon enhancement process has many subtleties, most of which are engineering details that we will not focused on here. However, a few of them are worth noting. First, lexicon enhancement is run only under specific circumstances; in particular, it is necessary to find at least two instances of a one-two alignment where the intersection of the possible parts of speech of the variable SL words must be non-empty. In other words: for each variable SL word, we collect all possible parts of speech from the lexicon and from the morphology. We then intersect these sets of possible parts of speech. For enhancement, we then only apply replacement rules to words of the part(s) of speech in this intersection. Second, we currently focus only on one-two (rather than one-three, etc.) alignments. This was done in order to control the complexity of the process, and as a safeguard against unbounded overgeneralization. Third, we currently only focus on suffixes. This indicates that much future work is possible in this area, and will be discussed at the end of this section. Finally, when English is the TL, it is not necessarily required to learn derivational morphology rules. Existing English morphology modules could be integrated with this approach in the future.

The lexicon enhancement was run on the full structural corpus. The goal of this module is to introduce as many new lexicon entries as possible. For this reason, we opted in favor of a certain amount of language-specific engineering in order to demonstrate how the algorithm works. This means that our experiment was no longer a fully automatic experiment, but that it was ensured that we would be able to introduce lexicon entries for 1) Hebrew

adverbs and 2) Hebrew comparative adjectives, as described below.

The three most important replacement rules that were learned from the training corpus seed are listed below. The rules are to be read as in the following example:

ADJ→ADJ by “”→ “ER”

is to be read as:

An `ADJ` can be transformed into an `ADJ` by replacing suffix “” with “ER”

The following are successful replacement rules that were inferred:

1. ADJ→ADJ by “”→“ER”, e.g., ‘TALL’→‘TALLER’
2. ADV→ADJ by “Y”→“IER”, e.g., ‘HAPPY’→‘HAPPIER’
3. N→ADV by “NESS”→“LY”, e.g., ‘FOOLISHNESS’→‘FOOLISHLY’
4. N→ADV by “CY”→“TLY”, e.g., ‘CONSISTENCY’→‘CONSISTENTLY’

The last two rules are used to introduce new adverb entries as in our example. The first two are used for Hebrew comparative adjectives: In Hebrew, adjectives in the comparative are expressed as ADJ ‘IWTR’ (literally ADJ ‘MORE’). In other words, the constant word in this example is ‘IWTR’, while the variable words in TL and SL are the adjectives, where the English adjective is in the comparative and the Hebrew adjective is in its base form.

Some rules that were inferred and can be classified as wrong are:

1. PREP→PREP by “TO”→“INTO”
2. ADV→N by “HASTE”→“QUICKLY”

Using this algorithm on all the manually enhanced structural corpus resulted in a total of 728 new rules. Below, we list several new lexical rules.

```
ADJ::ADJ |: ["$B&" "IWTR"] -> ["FULLER"]
ADJ::ADJ |: ["$Q@" "IWTR"] -> ["CALMER"]
ADJ::ADJ |: ["&LWB" "IWTR"] -> ["POORER"]
ADJ::ADJ |: ["&ML" "IWTR"] -> ["LABOURER"]
ADJ::ADJ |: ["XWLH" "IWTR"] -> ["SICKER"]
ADV::ADV |: ["$MX" "IWTR"] -> ["HAPPIER"]
ADV::ADV |: ["&SWQ" "IWTR"] -> ["BUSIER"]
ADV::ADV |: ["B" "@P$WT"] -> ["FOOLISHLY"]
ADV::ADV |: ["B" "@QSIWT"] -> ["CEREMONIOUSLY"]
ADV::ADV |: ["B" "@RWP"] -> ["CRAZILY"]
```

The conclusion is that the automatic process manages to capture some interesting new lexical items, but it is not perfect. For instance, 'HAPPIER', while a good translation of its Hebrew equivalent, is tagged as an adverb rather than an adjective.

The work described here is only the beginning of what could be in itself a larger research direction on lexicon enhancement, and in particular lexical enhancement for categorial variation. We wanted our approach to be as language-independent as possible; thus, the approach is completely automated, but can be seeded with language-specific information if such information is available. It could easily be applied to another language pair. The derivational morphology rules inferred here are only for those cases that were observed in the training data, i.e. for the seeds. In this way, as stated above, the algorithm is dependent on what it is seeded by. An interesting area for future research would be to automatically learn derivational morphology rules, in particular in bilingual setting and with the help of POS tags. As was said above, we currently only learn suffix replacement rules. This indicates that this project is only the beginning, and could be expanded to prefixes, circumfixes, and infixes as well, however with much increased complexity. For the purpose of this thesis, the current work has proven useful, as its goal was to show a solution to the given problem. For this reason, we will not develop this direction further for this thesis.

7.6.4 Generalization to Part-of-Speech Level

This section describes a special case that is both rare in our training data and subtle. We present here a solution to this special case. It should however be noted that this is merely a proof of concept, a proof that a solution can be found for the problem at hand. The solution provided here has no bearing on the overall system results and is mostly interesting from a theoretical perspective.

During Seed Generation, we raised lexical items to the POS level only if the they were aligned one-one *and* the two one-one aligned words did not appear in the dictionary with different parts of speech. This was done as a precaution: if there is a mismatch in POS, it is not desirable to propose a consistent transfer from one POS into another. For example, it is undesirable to have a rule that consistently translates any verb into an adjective simply after seeing one such example in the training data. Rather, we propose a more robust algorithm by which in such cases the rules retain the lexical items, and a raising to the POS level is proposed only after more evidence is observed.

For example, we observe in the training data that between Hebrew and English, English demonstrative pronouns, e.g. 'THIS', translates into equivalent Hebrew demonstrative pronouns, as made explicit with the following dictionary entries:

```
DEMPRO::DET ["ZH"] -> ["THIS"]
(
(X1::Y1)
((X1 GEN) = M)
((X1 NUM) = S)
)

DEMPRO::DET ["ZWT"] -> ["THIS"]
(
(X1::Y1)
((X1 GEN) = F)
((X1 NUM) = S)
)

DEMPRO::DET ["ALH"] -> ["THESE"]
(
(X1::Y1)
((X1 NUM) = P)
)
```

The reason for the POS mismatch here is that Hebrew demonstrative pronouns do not behave like other determiners and are thus tagged with a different label in the dictionary, while English demonstrative pronouns are tagged as determiners, because they act like ordinary determiners. Thus, when using the lexicon during rule learning, the following training examples result in the rules listed with them:

```
TL: this city
SL: H &IR H ZWT
Alignment: ((1,4),(2,2))
Type: NP
CoEmbeddingScore: 2
C-Structure:(<NP> (DET THIS-1) (N CITY-2))
```

results in

```
;;SL: H &IR H ZWT
;;TL: this city
NP::NP ["H" N "H" "ZWT"] -> ["THIS" N]
(
(X2::Y2)
)
```

and

```
TL: these cities
SL: H &IRIM H ALH
Alignment: ((1,4),(2,2))
Type: NP
CoEmbeddingScore: 2
C-Structure:(<NP> (DET THESE-1) (N CITIES-2))
```

results in

```
;;TL: H &IRIM H ALH
;;SL: these cities
NP::NP ["H" N "H" "ALH"] -> ["THESE" N]
(
(X2::Y2)
)
```

As a separate optional step before Compositionality Learning, we then inspect all learned rules and raise the lexicalized items to the POS level if we encounter more than one example of given POS labels translating into a specific different POS label. It would be desirable here to have a less heuristic cutoff. However, the number of training examples for this special case are *extremely* limited: in our training data, we only encountered the example involving the demonstrative pronouns and no other examples. For future work, it would be interesting to measure how much this extreme sparseness of training data is a function of the dictionary, as in our dictionary contains almost no instances of entries with different parts of speech for the Hebrew and the English word. It is not infrequent for a word to translate into a different part of speech. In our system, we are limited in catching and handling this phenomenon simply by the given dictionary. The algorithm that allows one-one aligned words of different POS to be generalized to the POS level can be represented in pseudocode as in Figure 7.8.

When applying this step to the example above, we obtain the following more general rules:

Generalization to POS level

For each rule r_i with one-one aligned words w_{TL} w_{SL} that are listed in the dictionary with different parts of speech $type_{w_{TL}}$ and $type_{w_{SL}}$, respectively

    If there is another rule r_j with a one-one aligned word pair w'_{TL} w'_{SL} with the same parts of speech as w_{TL} and w_{SL}, namely $type_{w_{TL}}$ and $type_{w_{SL}}$, respectively

        Replace w_{TL} with $type_{wTL}$ and w_{SL} with $type_{wSL}$ in rule r_i
        Replace w'_{TL} with $type_{wTL}$ and w'_{SL} with $type_{wSL}$ in rule r_j

    If there are any other rules r_k with a one-one aligned word pair w''_{TL} w''_{SL} with the same parts of speech as w_{TL} and w_{SL}, namely $type_{w_{TL}}$ and $type_{w_{SL}}$, respectively

        Replace w''_{TL} with $type_{wTL}$ and w''_{SL} with $type_{wSL}$ in rule r_k

Figure 7.8: Pseudocode for generalization to POS level.

```
;;SL: H &IR H ZWT
;;TL: this city
NP::NP ["H" N "H" DEMPRO] -> [DET N]
(
(X2::Y2)
(X4::Y1)
)

;;SL: H &IRIM H ALH
;;TL: these cities
NP::NP ["H" N "H" DEMPRO] -> [DET N]
(
(X2::Y2)
(X4::Y1)
)
```

7.6.5 Subtleties in Compositionality Learning

One-many and many-one alignments revisited

All many-one and one-many alignments that do not fall in the lexicon enhancement categories, in the compound category, or in the pro-drop category are not generalized to the POS level. This includes complex alignments such as

```
TL: the world 's famous dictators
SL: H DIQ@@WRIM H MPWRSMIM $L H &WLM
Alignment: ((1,1),(1,3),(2,7),(3,5),(4,4),(5,2))
Type: NP
CoEmbeddingScore: 2
C-Structure:(<NP> (<NP> (DET the-1)(N world-2))(GEN 's-3)
    (ADJ famous-4)(N dictators-5))
```

These cases cannot easily be handled in our system, because they are very rare and would be very susceptible to learning idiosyncrasies and thus overfitting to the training data. Therefore, we have chosen to not generalize from these cases, and leaving the words in question lexicalized. While this approach comes at the expense of some generality in the learned rules, it is nevertheless a safer choice. This is an instance of a trade-off between generality and overfitting to the training data. It would be interesting to explore how to approach these more complex cases with more training data available; however, this is not in the scope of this thesis.

There are, however, certain ways in which Compositionality Learning can overcome the problems posed by components that are not aligned one-one. During Seed Generation, words that are not aligned one-one in the training data are left lexicalized. During Compositionality Learning, it is not generally desirable to prevent not one-one aligned words from participating in a compositional element.

For example, if it were enforced that an unaligned word remain lexicalized and not participate in compositionality, the following seed rule

```
;;SL: $ HWA $R $IR
;;TL: THAT HE SANG A SONG
SBAR::SBAR [SUBORD PRO V N] -> [SUBORD PRO V "A" N]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
(X4::Y5)
)
```

would be generalized during Compositionality to:

```
;;SL: $ HWA $R $IR
;;TL: THAT HE SANG A SONG
SBAR::SBAR [SUBORD NP V N] -> [SUBORD NP V "A" N]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
(X4::Y5)
)
```

Clearly, the unaligned indefinite determiner 'A' would lower the level of compositionality that can be learned. By contrast, if allowing unaligned words to participate in compositionality, the following rule can be learned:

```
;;SL: $ HWA $R $IR
;;TL: THAT HE SANG A SONG
SBAR::SBAR [SUBORD S] -> [SUBORD S]
(
(X1::Y1)
(X2::Y2)
)
```

In order to develop a clearer picture of the problem, we categorize not one-one alignments as follows:

1. **UnalignedTL:** Unaligned TL indices
2. **UnalignedSL:** Unaligned SL indices
3. **OneToMany:** one-many word alignments
4. **ManyToOne:** many-one word alignments
5. **ManyToMany:** a group of TL indices aligned to a group of SL indices
6. **DiscontSL:** a TL constituent that aligns to two or more coherent SL indices that are interrupted by another set of indices

Most of these categories are straightforward, but the last one warrants an example for clarification. An example of DiscontSL is:

```
TL: a dispute with the school board
SL: SKSWK &M W&D BIT H SPR
Alignment: ((2,1),(3,2),(4,5),(5,4),(5,6),(6,3))
Type: NP
CoEmbeddingScore: 4
C-Structure:(<NP> (<NP> (DET a-1)(N dispute-2))
    (<PP> (PREP with-3)(<NP> (DET the-4)(N school-5)
    (N board-6))))
```

In this case, the contiguous component 'SCHOOL BOARD' is interrupted by the Hebrew word 'H'.

It is interesting to examine what impact these special cases have on the learned rules. In essence, two things can happen with not one-one aligned indices. They can either remain lexicalized, or else they can be merged into a higher-level constituent. In the following, we discuss some training examples and how the learning algorithms handle them.

The training example with an unaligned TL index

```
TL: the jury did not elaborate
SL: XBR H MW$B&IM LA HTDINW
Alignment: ((1,2),(2,1),(2,3),(4,4),(5,5))
Type: S
CoEmbeddingScore: 3
```

```
C-Structure:(<S> (<NP> (DET the-1)(N jury-2))
    (<AUX> (V did-3))(NEG not-4)
    (<VP> (V elaborate-5)))
```

results in the following rule:

```
;;SL: XBR H MW$B&IM LA HTDINW
;;TL: THE JURY DID NOT ELABORATE
;;C-Structure:(<S> (<NP> (DET the-1)(N jury-2))
    (<AUX> (V did-3))(NEG not-4)
    (<VP> (V elaborate-5)))
S::S [NP ADV V] -> [NP "DID" ADV V]
(
(X1::Y1)
(X2::Y3)
(X3::Y4)
)
```

In this rule, the auxiliary is left lexicalized, which can be useful at runtime, because it capture the divergence that in Hebrew, negated verbs are not split into a verb and an auxiliary like in English. Therefore, it is useful to introduce the auxiliary as a lexicalized item.

Consider another example. The training example

```
TL: a meter long
SL: B AWRK M@R
Alignment: ((2,3),(3,1),(3,2))
Type: ADJP
CoEmbeddingScore: 3
C-Structure:(<ADJP> (<NP> (DET a-1)(N meter-2))
    (ADJ long-3))
```

produces the following rule:

```
;;SL: B AWRK M@R
;;TL: A METER LONG
;;C-Structure:(<ADJP> (<NP> (DET a-1)(N meter-2))
    (ADJ long-3))
ADJP::ADJP ["B" "ARK" NP] -> [NP "LONG"]
(
;(X1::Y2)
```

```
;(X2::Y2)
(X3::Y1)
)
```

In this case, the NP is correctly captured and generalized to, whereas the adjective remains lexicalized. The lexicon enhancement cannot cover this example, because there was not enough evidence of English adjectives translating into the preposition 'B' and a noun. Possibly a larger training set would ameliorate this limitation. With the given training set, the best solution is for the not one-one words to remain lexicalized, so as to not overgeneralize (for instance to PREP N → ADJ).

Further, the training example

```
TL: before they arrived
SL: LPNI $ HM HGI&W
Alignment: ((1,1),(2,3),(3,4))
Type: SBAR
CoEmbeddingScore: 4
C-Structure:(<SBAR> (PREP before-1)
    (<S> (<NP> (PRO they-2))(<VP> (V arrived-3))))
```

resulted in the following rule:

```
;;SL: LPNI $ HM HGI&W
;;TL: BEFORE THEY ARRIVED
;;C-Structure:(<SBAR> (PREP before-1)
    (<S> (<NP> (PRO they-2))(<VP> (V arrived-3))))
SBAR::SBAR [PREP "$" S] -> [PREP S]
(
(X1::Y1)
(X3::Y2)
)
```

This training example captures the important translation mismatch that Hebrew prepositions are often followed by the conjunction '$' to introduce a subordinate clause.

No examples were found in the training data of an unaligned SL word being integrated in a higher-level constituent. For example, the training example

```
TL: the boy ate the apple
SL: H ILD AKL AT H TPWX
Alignment: ((1,1),(2,2),(3,3),(4,5),(5,6))
Type: S
CoEmbeddingScore: 4
C-Structure:(<S> (<NP> (DET the-1)(N boy-2))
        (<VP> (V ate-3)(<NP> (DET the-4)(N apple-5))))
```

results in the following rule:

```
;;SL: H ILD AKL AT H TPWX
;;TL: THE BOY ATE THE APPLE
;;C-Structure:(<S> (<NP> (DET the-1)(N boy-2))
        (<VP> (V ate-3)(<NP> (DET the-4)(N apple-5))))
S::S [NP V "AT" NP] -> [NP V NP]
(
(X1::Y1)
(X2::Y2)
(X4::Y3)
)
```

This rule is learned even when Maximum Compositionality is assumed. This is because the 'AT' could only be part of a higher-level VP. VP, however, is not on the list of constituents of interest, so no VPs are produced, and the 'AT' must remain lexicalized.

To give one example of the category DiscontSL, the training example

```
TL: a dispute with the school board
SL: SKSWK &M W&D BIT H SPR
Alignment: ((2,1),(3,2),(4,5),(5,4),(5,6),(6,3))
Type: NP
CoEmbeddingScore: 4
C-Structure:(<NP> (<NP> (DET a-1)(N dispute-2))
    (<PP> (PREP with-3)(<NP> (DET the-4)(N school-5)
    (N board-6))))
```

produced the following rule under Maximum Compositionality:

```
;;SL: SKSWK &M W&D BIT H SPR
;;TL: A DISPUTE WITH THE SCHOOL BOARD
;;C-Structure:(<NP> (<NP> (DET a-1)(N dispute-2))
```

```
    (<PP> (PREP with-3)
    (<NP> (DET the-4)(N school-5)(N board-6))))
NP::NP [NP PP] -> [NP PP]
(
(X1::Y1)
(X2::Y2)
)
```

Despite the fact that the indices corresponding to the English component 'SCHOOL BOARD' are discontinuous, the Compositionality algorithm manages to abstract away from this, as a higher-level component 'THE SCHOOL BOARD' translates into a contiguous chunk of SL indices.

Discussion: Recovering from Noise in the Training Data

In certain cases, the learning algorithms can help the system overcome noisy training data. Consider the following example:

```
TL: the press and its readers
SL: H &TWNWT W QWRIAI H
Alignment: ((1,1),(2,2),(4,5),(5,4))
Type: NP
C-Structure:
(<NP>
  (<NP> (DET the-1)(N press-2))
  (CONJ and-3)
  (<NP> (POSS its-4)(N readers-5)))
```

The translation in Hebrew does not translate 'ITS' properly (the aligned word 'H' is the definite singular determiner in Hebrew). Thus, a flat rule would incorrectly reflect how a possessive translates into Hebrew. Luckily, this problem can be 'hidden' when the learning algorithm is run with the assumption of Maximum Compositionality. It then infers correctly the following higher-level rule:

```
;;SL: H &TWNWT W QWRIAI H
;;TL: THE PRESS AND ITS READERS
NP::NP [NP "W" NP] -> [NP "AND" NP]
(
(X1::Y1)
(X3::Y3)
)
```

(Note that 'AND' and 'W' remain lexicalized because the translator did not align them, although they are actually translations of each other).

While the Compositionality algorithms were not designed specifically to overcome noise in the training data, they were in fact designed to learn maximally compositional rules, which has the potential side-effect of hiding noise that is contained in the lower-level constituents.

7.6.6 Structural Grammar Enhancement

The previous sections discussed how rules are learned from training examples, and how complex cases are handled. This section is different in the sense that it uses an existing learned grammar to infer further transfer rules. As we will describe below, structural grammar enhancement takes in a learned grammar that is as large as possible, and infers additional rules from it.

Consider the following pair of component sequences:

```
NP::NP [Det Adj Adj N] -> [Det N Adj Adj]
NP::NP [Det Adj N] -> [Det N Adj]
```

To a human grammar designer, these two sequences are not only similar, but indicate a possible generalization. The underlying rule for both of the above sequences could be

```
NP::NP [Det Adj+ N] -> [Det N Adj+]
```

In other words, the two sequences above may lead to conclude that for more than two adjectives, the language pair behaves similarly. However, if

```
NP::NP [Det Adj Adj Adj N] -> [Det N Adj Adj Adj]
```

was not observed in the training data, but such a structure was encountered in the test data, the grammar would not be able to cover this structure at run-time. The goal of the work described here is to infer automatically when the grammar can be generalized in this way. Although in practice a test sentence cannot contain an infinite number of adjectives in succession, and in fact the number of adjectives in succession is bounded by a small number, such generalizations are desirable simply because the training data may not reach those practical limits. In the evaluation section we report that the generalized rules were applied at run-time, thus validating the approach in practice, not only in theory.

We now give a brief description of the steps involved in obtaining such generalizations. The subsequent sections will explain each of the steps in greater detail.

1. **Learned Grammar.** The original learning algorithm extracts a number of transfer rules from a bilingual corpus. The set of learned rules, the grammar, is the input to the grammar enhancement module.

2. **Pairs of Rules.** The first step in enhancing a learned transfer grammar is to consider all pairs of rules *of the same type*. All rule types are considered for expansion. The reason that we want to find pairs of rules as in the example above is that we want to maximize the evidence from the data before we propose a generalized rule. We do not want to introduce an X+ whenever we observe an X component in the data, because an observation of X in a particular context provides very little evidence that this component should be generalized. Especially in the face of very limited data, as in our system, it is important to safeguard against drawing generalizations too quickly.

3. **Minimal Alignment.** In this step, the algorithm minimally aligns the component sequences of the rule pair, for the source and target languages independently. Minimal alignment aligns the sequences with the minimal number of gaps.

4. **Enhancement Criteria.** Minimal alignment always finds a solution, even for pairs of rules whose component sequences are not in the least similar. For this reason, a pair of rules and their minimal alignment must fulfill a number of enhancement criteria before it can be generalized. These enhancement criteria essentially check whether the rule pair is an example such as the one described above in this section.

5. **Generalization.** Generalization works by adding rules to the grammar in order to allow for an infinite sequence of the generalized constituent. Generalization is done by introducing another level of compositionality and a new non-terminal into the grammar.

Pairs of Rules

Starting with an existing transfer grammar, we consider all pairs of rules that are of the same type. As described above, each rule in our grammar is labeled with a top node that essentially indicates the rule's role in the grammar, e.g. PP or S. Two rules are only considered for generalization if they are

of the same type, because we only want to generalize two rules if they are sufficiently similar. The generalization produces a rule that 'explains' both of the underlying rules. Two rules of different types cannot be explained by just one rule, as they play different roles in the grammar (e.g. combine compositionally with a different set of rules).

Minimal Alignment of Rules

In order to compare pairs of rules, we minimally align the component sequences using an algorithm that is routinely used in DNA sequence analysis, the Needleman-Wunsch algorithm (Needleman & Wunsch, 1970). This algorithm relies on dynamic programming to find an alignment between two sequences that introduces a minimum number of gaps. Each gap is penalized by lowering the score of the alignment. The algorithm proceeds by filling the two-dimensional table formed by the two candidate sequences, starting from the top left corner. Each entry in the table receives a score, indicating the goodness of the alignment sequence up to this point in the table. The highest-scoring sequence is then obtained by backtracking through the table. Table entries are filled as follows:

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, x_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

d is the gap penalty, in our case $d = 1$ and $s(x_i, x_j)$ is the score of an alignment, in our case $s = 100$ if the labels are the same $s = -100$ otherwise. In order to get reasonable alignments, it is important that it is penalized highly to align two labels that are not the same. With a penalty of 100, as used here, two different labels will never be aligned to each other, as desired. An intuitive explanation of this algorithm is that at each step in the alignment, one can either skip a word in the first sequence, skip a word in the second sequence, or else move forward in both sequences. All of these possibilities are associated with a certain score that reflects the contribution of this move to the overall alignment. As the goal is to introduce a minimum number of gaps (and gaps are penalized by lowering the score), the algorithm always chooses from the three possibilities the one with the highest score. This means in effect that each table entry receives the highest score of all possible alignments leading to it. Table 7.1 shows the application of the algorithm to the sequences in the example above. The algorithm proceeds through the table, filling it with the maximally possible scores.

		Det	Adj	Adj	N
	0	-1	-2	-3	-4
Det	-1	100	99	98	97
Adj	-2	99	200	199	198
N	-3	98	199	198	299

Table 7.1: Complete minimal alignment table.

When backtracking through the table to find the highest-scoring alignment, we start from the bottom right and proceed up, left, or diagonally up left, always choosing the step with the maximal score, until we reach the top-left corner. A move diagonally up means 'scanning' a symbol from both sequences, whereas moving up or left implies that a gap is introduced in the final alignment.

Since the table was filled by essentially capturing all the scores up to this point, backtracking by always choosing the maximal step is not greedy. For the example in Table 7.1, the resulting minimal sequence is

Det	Adj	Adj	N
Det	Adj	_	N

The minimal alignment algorithm is performed on all pairs of rules of the same type. Each rule has a SL and a TL component sequence; they are aligned separately, so that for each rule pair we obtain a SL minimal alignment and a TL minimal alignment.

Grammar Enhancement Criteria

As was said above, the Needleman-Wunsch algorithm always finds a solution. All pairs of sequences can be minimally aligned, even if they are not similar whatsoever. This means that we must interpret the output of the minimal alignment algorithm to determine if the pair of rules is actually sufficiently similar to propose a generalization. This is done by applying several 'tests' to the minimal alignments of source and target language component sequences as well as on the entire rules. The tests are crucial, so that the grammar is not overgeneralized with invalid rules. In order to result in a generalization, a pair of minimally aligned rules must fulfill *all* of the following criteria:

For example, the sequences above would fulfill all of these criteria, since one rule has and adjective on both sides that the other rule does not have,

1. **The minimal alignment must contain exactly one gap each on the source and the target language side.** This constraint is introduced in order to enforce that the rules are sufficiently similar. Each newly introduced rule is designed to generalize over exactly one constituent.

2. **Both gaps must be found in one and the same rule.** The constituent that is to be generalized over should be present in both languages for one rule, and not present in both languages for the other rule. This constraint ties in with the previous one. In essence, we want to ensure that the longer rule (i.e. the one without gaps) includes a constituent on both sides that is missing in the shorter rule, such that the constituent in both languages translate into each other.

3. **The source language component sequence must have a constituent adjacent to the gap that is of the same type as the consistent corresponding to the gap.** This will ensure that one of the rules has one instance of the constituent, whereas the other has two instances in sequence. If this criterion is met, we essentially generalize from seeing one and two instances to potentially infinitely many instances.

Figure 7.9: Enhancement criteria for structural grammar enhancement.

and the shorter rule has another adjective adjacent to the gap.

Producing Additional Rules

For each pair of rules that is minimally aligned and fulfills the grammar enhancement criteria, three new rules are added to the grammar. These three rules in effect implement the regular expression C+, where C is the constituent that is being generalized. A unique constituent label is introduced into the grammar, in place of the gaps in the shorter rule. Two additional rules of type C are then introduced, allowing C to turn into one terminal of the generalized constituent and another C non-terminal, or else to produce just one terminal. In the case of our example, the three introduced rules will look as follows:

```
NP::NP [Det Adj C N] -> [Det N Adj C]

C::C [C Adj] -> [C Adj]

C::C [Adj] -> [Adj]
```

Now that we have described all parts of the enhancement algorithm, we can summarize it in pseudocode as in Figure 7.10.

Discussion of Additional Rules

In order to assess the effectiveness of the proposed approach, we learned an initial context-free grammar from about 470 sentences and phrases. The original grammar contained 407 unique rules. The context-free grammar was then fed into the enhancement algorithm, producing an additional 24 unique rules. An example of a series of newly added rules to the grammar can be seen below:

```
NP::NP [ADJ C0 N] -> [N ADJ C0]
(
(X1::Y3)
(X2::Y1)
(X3::Y2)
)

NP::NP ["A" ADJ C0 N] -> [N ADJ C0]
```

Structural Grammar Enhancement

For all pairs of rules r_1, r_2 of the same type $type_i$

Find a minimal alignment of the component sequences

If the rule r_1 and r_2 fulfill enhancement criteria as specified in Figure 7.9

Then r_2 contains one gap in the SL and TL component sequences in the minimal alignment, where r_1's component sequences contain a component of type C, and where r_1 and r_2 contain a component $adjac$, that is adjacent to the SL gap and that is of the same type as C.

Introduce three additional rules that implement the regular expression $C+$, where C is the generalized constituent:

$type_i$::$type_i$ $[\ldots\ adjac\ C\ \ldots] \rightarrow [\ldots\ adjac\ C\ \ldots]$
C::C $[C\ adjac] \rightarrow [C\ adjac]$
C::C $[adjac] \rightarrow [adjac]$

Figure 7.10: Pseudocode for structural grammar enhancement.

```
(
(X2::Y2)
(X3::Y3)
(X4::Y1)
)
```

These rules apply to indefinite NPs. In Hebrew indefinite NPs are not marked with a determiner, while in English they are in some cases (e.g. 'A TALL MAN', i.e. singular indefinite, vs. 'TALL MEN', i.e. plural indefinite). This is reflected by an ambiguous translation in the two rules above. The algorithm proposed in this paper detected that in both cases, the NP can contain an arbitrary number of adjectives, as is reflected in the rules. A few things are interesting to note about this series of rules. The statistical decoder will deal with the ambiguity inherent in these rules, and will extract the appropriate partial translations to combine them into a full translation. Future work will aim at introducing a set of constraints into the rules, so that their applicability or possible output can be restricted and thus relieve the burden on the decoder.

The following examples are generalized rules learned for noun compounding. The first three, a straightforward reduplication of the noun is applies on both sides. The fourth rule is particularly interesting, because it captures the phenomenon that definite noun compounds in Hebrew are marked for definiteness only before the last noun. This example also illustrates the power of the alignment algorithm: a more simple algorithm that would simply scan the original grammar for cases where components are reduplicated on both sides would not capture this case.

```
NP::NP [N C2] -> [N C2]
(
(X1::Y1)
(X2::Y2)
)

NP::NP [ADJ N C2] -> [N C2 ADJ]
(
(X1::Y3)
(X2::Y1)
(X3::Y2)
)
```

```
NP::NP  ["A" ADJ N C2] -> [N C2 ADJ]
(
(X2::Y3)
(X3::Y1)
(X4::Y2)
)

NP::NP [DET N C2] -> [C2 DET N]
(
(X1::Y2)
(X2::Y3)
(X3::Y1)
)
```

Although the algorithm does capture interesting phenomena, it is limited in certain ways, and future research could address these limitations. Consider the following rule in the original grammar:

```
;;SL: H B&IH H $NIH H GDWLH
;;TL: THE SECOND BIG PROBLEM
NP::NP ["THE" ADJ ADJ N] -> ["H" N "H" ADJ "H" ADJ]
(
(X2::Y4)
(X3::Y6)
(X4::Y2)
)
```

The current generalization algorithm is not able to capture the fact that in a definite Hebrew noun phrase, each adjective is marked for definiteness separately. In this case, the algorithm would have to detect that the original rule actually represents a more complex reduplication.

Another limitation of our current algorithm is that although it maximizes support from the original grammar before introducing a generalization, this might still not be enough in all cases. There are currently no safeguards against generalizations that are supported only by noise in the data, since we generalize after seeing merely one rule with reduplication and one rule without reduplication. Future work will address this in the context of scoring rules based on their translation power on the training set. We have done some initial investigation on this topic, as it can be applied to all learned rules, not only the generalized ones. In this framework, rules will be scored

based on whether they can correctly translate sentences (or parts of sentences) other than the ones they were derived from. Future work could address this issue in detail.

7.7 Applying Quality Criteria

In all previous sections, we aimed at extracting the maximally compositional rules from the training data. We applied certain safeguards against overgeneralization, but the overall goal was to strive towards high generality of the rules. By contrast, in this section we introduce two filters that are applied to the rules, so that the learned grammars will not contain obviously overly general rules, and so that the rules abide by the axioms put forth in section 7.2.4. Any learned rule must pass these two filters in order to be included in the final grammar.

7.7.1 Quality Criterion 1: Checking for Boundary Crossings

The alignments give us sets of indices that group together, denoted *component sets*. For TL, these groups consist of all the indices that are covered by an interior parse node. From the alignments we can infer the corresponding sets for SL. The component sets allow us to gain insight into how well the TL structure is preserved when being translated into SL.

As discussed earlier, in some training examples the TL component sets do not align to coherent sets on the SL side. This can be detected by checking for so-called *Boundary Crossings*, where a SL component set includes an alignment not only to words in the corresponding TL component set, but also to one or more words in a different TL component set. This is best illustrated with an example:

```
TL: instead of a book
SL: BMQWM SPR
Alignment: ((1,1),(2,1),(4,2))
Type: PP
C-Structure:
(<PP>
  (ADV instead-1)
  (<PP> (PREP of-2)(<NP> (DET a-3)(N book-4))))
```

This is an example of the advanced case of DiscontSL as was described above. In the previous example, the DiscontSL effect was masked, because

the discontinuity was embedded within a higher-level structure for which a compositional element could be introduced. In this example, however, this is not the case, so that a different approach must be taken.

It can be seen that the English structure is not preserved in Hebrew, as is indicated by the word alignments and the boundary crossing between the component set corresponding to the ADV and component set corresponding to the PP. In the absence of an SL parser, we cannot propose any structure, so that falling back onto the lexical items is the best solution, as reflected in the learned rule:

```
;;SL: BMQWM SPR
;;TL: INSTEAD OF A BOOK
PP::PP ["BMQWM" "SPR"] -> ["INSTEAD" "OF" "A" "BOOK"]
(
;(X1::Y1)
;(X1::Y1)
;(X2::Y4)
)
```

This example can be explained better with a diagram, as in Figure 7.11. The PP 'OF A BOOK' is 'split' in Hebrew, because the preposition aligns to a word that is also aligned with a word outside the prepositional phrase.

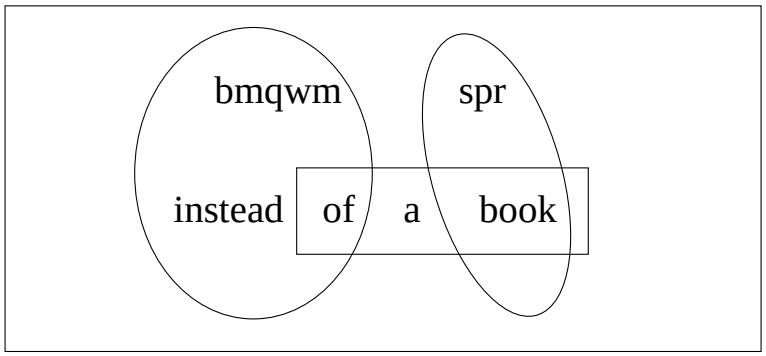

Figure 7.11: Diagrammatic representation of boundary crossing.

We cannot reach a higher level of compositionality in such a case, because a TL constituent does not align to a coherent SL component, where no other TL component aligns to this SL chunk.

If we proposed compositional elements, the SL word 'BMQWM' would participate in more than one component in the resulting component sequence

(the PREP and the NP). This would however violate the last axiom, which states that each word must participate in *exactly* one component.

7.7.2 Quality Criterion 2: No Unaligned Constituents

In the following training pair, one of the constituents is completely untranslated and thus unaligned: 'IN EFFECT' remains untranslated, presumably because the training example does not easily translate into Hebrew.

```
TL: in fact and in effect
SL: L M&$H
Alignment: ((1,1),(1,2),(2,1),(2,2))
Type: PP
C-Structure:
(<PP>
  (<PP> (PREP in-1)(<NP> (N fact-2)))
  (CONJ and-3)
  (<PP> (PREP in-4)(<NP> (N effect-5))))
```

In the case where a complete constituent, not simply a single word, remains untranslated, we must assume that the training example is noisy. In such a case, we do not produce any transfer rule at all.

7.8 Results

7.8.1 Discussion of Learned Rules

Comparison of Compositional and Non-Compositional Rules

In this section, we present a number of rules that were automatically learned using our system. The training corpus was the structural elicitation corpus of 120 examples.

Consider the following three rules. The first one represents the rule that was learned using Seed Generation only. This is followed by the corresponding compositional rules that were learned with and without the assumption of Maximum Compositionality, respectively.

The seed rule looks as follows:

```
;;SL: $ B TWK H M&@PH HIH $M
;;TL: THAT INSIDE THE ENVELOPE WAS A NAME
SBAR::SBAR [SUBORD "B" "TWK" DET N V N] ->
```

```
                [SUBORD "INSIDE" DET N V "A" N]
(
(X1::Y1)
(X4::Y3)
(X5::Y4)
(X6::Y5)
(X7::Y7)
)
```

It can be seen the rule is flat, which is necessarily the case when Compositionality Learning is not used. Further, the rule generalizes some of the word indices to the POS level, while other words are left lexicalized. For example, 'INSIDE' is left lexicalized because it aligns to two Hebrew words, whereas 'A' is left lexicalized because it is unaligned. However, several words are generalized to the POS level. This results in a rule that generalizes well beyond the specific training example it was derived from.

When Compositionality Learning is turned on, the following rule is learned *without* the assumption of Maximum Compositionality:

```
;;SL: $ B TWK H M&@PH HIH $M
;;TL: THAT INSIDE THE ENVELOPE WAS A NAME
SBAR::SBAR [SUBORD "B" "TWK" NP V N] ->
        [SUBORD "INSIDE" NP V "A" N]
(
(X1::Y1)
(X4::Y3)
(X5::Y4)
(X6::Y6)
)
```

This rule generalizes 'THE ENVELOPE' to an NP. This means that a rule learned in a previous learning step can translate the chunk 'H M&@PH' into 'THE ENVELOPE'. The rule is more general than the one learned with Seed Generation only, as any NP can now plug into the slot that is represented by NP in the rule.

Finally, we compare the previous two rules to the compositional rule that was learned under the assumption of Maximum Compositionality:

```
;;SL: $ B TWK H M&@PH HIH $M
;;TL: THAT INSIDE THE ENVELOPE WAS A NAME
```

```
SBAR::SBAR [SUBORD PP V NP] -> [SUBORD PP V NP]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
(X4::Y4)
)
```

This rule is again more general than the rule that was learned without the assumption of Maximum Compositionality. The generalization power of this rule is very high, as it can apply to any subordinate clause of the given structure, and is not lexically bound. Note that the unaligned 'A' has now been abstracted away from, as it is part of an NP.

As another example, the following is a rule produced using Seed Generation only:

```
;;SL: RQ AM H RKBT TGI&
;;TL: ONLY IF THE TRAIN ARRIVES
SBAR::SBAR [ADV SUBORD DET N V] -> [ADV SUBORD DET N V]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
(X4::Y4)
(X5::Y5)
)
```

In this case, the rule produced by Seed Generation alone is quite general. It still cannot combine with any other grammar rules, as it is not compositional. It is, however, general in the sense that it is no longer lexically bound.

The corresponding compositional rule without assuming Maximum Compositionality is:

```
;;SL: RQ AM H RKBT TGI&
;;TL: ONLY IF THE TRAIN ARRIVES
SBAR::SBAR [ADV SUBORD NP V] -> [ADV SUBORD NP V]
(
(X1::Y1)
(X2::Y2)
```

```
(X3::Y3)
(X4::Y4)
)
```

It can be seen that the compositional rule is more general than the corresponding seed rule because it no longer requires that the subject of the subordinate clause be a simple NP of the form 'DET N'. This means that the rule can apply to a larger number of structures at run-time.

Finally, running Seed Generation in combination with the Compositionality algorithm while assuming Maximum Compositionality is:

```
;;SL: RQ AM H RKBT TGI&
;;TL: ONLY IF THE TRAIN ARRIVES
SBAR::SBAR [ADVP SUBORD S] -> [ADVP SUBORD S]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
)
```

This rule correctly recognizes that in fact we are dealing with a subordinate clause that consists of a subordinating conjunction and a full sentence. Clearly, this rule is more general than the previous two, and manages to capture more of the structure of the two languages.

Comparison of learned rules with Maximum Compositionality and learned rules without Maximum Compositionality

One example of the comparison of learned rules when Maximum Compositionality is assumed versus when it is not assumed is as follows. The rule without assumption of Maximum Compositionality is:

```
;;SL: $&WT H &RB LA TMID BWLW B AWTW AWPN
;;TL: THE EVENING WAS NOT ALWAYS SPENT IN THE SAME WAY
S::S ["$&WT" NP ADV ADV V PREP ADJ N] ->
        [NP "WAS" ADV ADV V PREP "THE" ADJ N]
(
(X2::Y1)
(X3::Y3)
(X4::Y4)
(X5::Y5)
```

```
(X6::Y6)
(X7::Y8)
(X8::Y9)
)
```

Assuming Maximum Compositionality results in the following rule:

```
;;SL: $&WT H &RB LA TMID BWLW B AWTW AWPN
;;TL: THE EVENING WAS NOT ALWAYS SPENT IN THE SAME WAY
S::S ["$&WT" NP ADV ADVP V PP] -> [NP "WAS" ADV ADVP V PP]
(
(X2::Y1)
(X3::Y3)
(X4::Y4)
(X5::Y5)
(X6::Y6)
)
```

Clearly, the rule learned under the assumption of Maximum Compositionality captures the structure of the training example better, while the other rule is left mostly lexicalized.

The key to understanding this example is that the learning algorithm often has few 'previously learned rules' that can be used for testing the validity of compositionality.

7.8.2 Automatic Evaluation Results

Table 7.2 summarizes the automatic evaluation results for compositional grammars. We list several results here, which warrants some explanation: Aside from results for no grammar and the manual grammar as above, we repeat for convenience the results for rules learned during Seed Generation ('Learned Grammar (SeedGen)'). We then list the results for two compositional grammars, without and with the assumption of Maximum Compositionality ('Learned Grammar (Compos)' and 'Learned Grammar (MaxCompos)', respectively).

The above results indicate that an improvement in score can be observed for the compositional rules. The results are more highly statistically significant than the Seed Generation results, with the METEOR scores being less significant.

The increase in score over Seed Generation indicates the generalization power of rules that can combine with each other to cover unseen structures.

Grammar	BLEU	ModBLEU	METEOR
No Grammar	0.0255	0.0910	0.2681
Manual Grammar	0.0713	0.1209	0.3204
Learned Grammar (SeedGen)	0.0281	0.0969	0.2786
Learned Grammar (Compos)	0.0346	0.0998	0.2819
Learned Grammar (MaxCompos)	0.0344	0.0995	0.2811

Table 7.2: Automated evaluation results for Seed Generation and Basic Compositionality on test set 1. 'MaxCompos' stands for a run that assumes Maximum Compositionality.

Comparison	BLEU	ModBLEU	METEOR
SeedGen	[-0.0073,0.0018]	[-0.0149,0.0026]	p=0.112
Compos	[-0.0321,0.000]	[-0.0164,-0.020]	p=0.084
MaxCompos	[-0.0312,0.000]	[-0.0162,-0.0017]	p=0.108

Table 7.3: Seed Generation only and Basic Compositionality confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline.

The two approaches to Compositionality, with and without the assumption of Maximum Compositionality, perform about equally well, and both provide an improvement over the baseline. It is not clear a priori which one should be used in practice. Maximum Compositionality has two clear advantages: 1) it produces more human-readable rules, as was explained in section 7.8.1 above, and 2) grammars can be trained faster because of the smaller training-time complexity of the Maximum Compositionality (as discussed in section 7.5). The algorithm without the assumption of Maximum Compositionality, on the other hand, has the advantage that at run-time, the lattices are smaller, resulting in faster run-time performance. It should be decided on a case-by-case basis which approach to Compositionality Learning should be used.

It should however also be noted that unconstrained compositional rules are very general and combine with each other readily, leading to large lattices. The goal of the following chapter is to attack this issue by amending the rules with unification constraints. From a practical perspective, we hope to bound the generality of the rules, so that they apply to fewer contexts.

Again, it can be observed that the manual grammar outperforms the

learned grammar by a considerable margin. This can at least in part be attributed to the fact that test set 1 was used as a development set for the manual grammar. In chapter 9, we test the system on a different test set that is unseen both for the manual and the learned grammar, and show that the margin between the two diminishes.

Chapter 8

Learning Unification Constraints

8.1 Introduction

The goal of the two previous chapters was to learn a set of context-free translation rules. Context-free rules are appealing because they are easy to grasp and apply. Furthermore, context-free rules provide the highest level of generalization over the training data. It implies that the rules can be used to maximally cover unseen data at run-time, given the observed training data. This is useful especially when learning from a very small corpus, when a large number of structures, or combinations thereof, are not observed.

On the other hand, context-free rules can overgeneralize. For example, certain rules should only apply under a set of specific circumstances, e.g. only to definite NPs. It is not straightforward to capture such kind of contextual information in context-free rules alone. For this reason, this chapter aims at refining the context-free rule with *unification constraints* or simply *constraints*. Unification constraints can either 1) limit their applicability to certain contexts (thereby limiting parsing ambiguity), 2) ensure the passing of feature values from source to target language (thereby limiting transfer ambiguity), or 3) disallow certain target language outputs (thereby limiting generation ambiguity). The constraints are automatically learned from data, using morphological information for both languages.

Unification constraints are theoretically very appealing, because they allow us to capture the appropriate context for a given rule. It should however be noted that the statistical decoder is very strong at selecting the best partial translations to form a full translation. This is the case even when

the partial translations were produced by a context-free grammar and thus contain a large number of ambiguities. This means that limiting ambiguity with unification constraints can only be useful to translation quality if the constraints eliminate arcs from the lattice that are 1) incorrect translations and 2) would have been chosen by the decoder. In practice, this is hard to achieve because of the strength of the decoder. The practical goal of this chapter is thus to limit ambiguity, thus reducing run-time, while not eliminating any or many 'good' arcs from the lattice. Whether this goal is achieved can be measured by comparing translation quality with and without constraints, and by measuring the run-time of the system with and without constraints. We provide both measures in the evaluation section 8.2 below.

We have given several motivations for learning unification constraints. Let us now briefly return to another discussion. In section 3.2 above we have argued why structural learning and Unification Constraint Learning should be kept separate. In addition to the reasons listed there (facilitating the design of the algorithm, more flexibility in the settings of the algorithm, optionality of the constraints), there is another argument in favor of learning unification constraints in a separate step: the goal of structural learning is to accomplish maximal generalization over the training data. For example, if a PP with an embedded NP is observed, the embedded NP is recognized in the rule as a compositional element if possible. More generalization can be accomplished if this is done without constraints. Consider the following illustrative example, and suppose that constraints were added to rules at the same time as the structure was determined:

```
NP::NP [DET N] -> [DET N]
(
(X2 NUM = S)
(Y2 NUM = S)
)

PP::PP [PREP DET N] -> [PREP DET N]
(
(X3 NUM = P)
(Y3 NUM = P)
)
```

Given that the NP rule applies only to singular nouns, the embedded NP in the PP rule could not be recognized and would not be generalized

to the NP level. It is, however, desirable to generalize to the NP level in this case, so that different types of NPs can apply within the PP rule. There are two ways to overcome this problem: 1) run the system with the assumption of Maximum Compositionality, and 2) ignore the constraints while learning structure. If the second option is chosen, the learning modules are once again separate. We thus chose to separate the learning modules, even if the assumption of Maximum Compositionality is made. This solution provides a greater amount of flexibility in the settings in which the rule learner can be run, and it has no bearing on the system behavior if Maximum Compositionality is in fact assumed.

Having said that, there are in fact special cases where interleaving the learning phases could result in superior rules. This case will be discussed in the Future Work section (Section 10.3).

Before we delve into the theory of unification constraints, a brief reminder on notation: the following refer to the same entities: **Source Language (SL), minor language, and x-side**. Similarly, the following refer to the same: **Target Language (TL), major language, and y-side**. In our case, Hebrew and Hindi are the SLs in the Hebrew→English and Hindi→English systems, respectively, and English is the TL. These terms will be used as appropriate in context. For example, when referring to rule parts, we often use the terms x-side and y-side. When talking about translation in general, the terms SL and TL are often more appropriate.

8.2 Review of Unification Constraints

Constraints were briefly described in section 3.2. They are equations that can be added to parsing, generation, or translation rules in order to specify context or pass feature values. Unification constraints are often used in LFG-style grammars (Bresnan, 2001).

In this section, we will only discuss constraints to the level that is necessary to understand the work described in this document. For further information on unification, pseudo-unification, and constraints, the interested reader should refer to (Shieber, 1986). The transfer engine used in our system performs pseudo-unification. This means that the result of a unification is passed to the left-hand side of the constraint equation. If the feature was not marked on the left-hand side, the feature value is merely passed from the right-hand side to the left-hand side.

Some constraints are referred to as *head*-passing constraints. A head can informally be described as the 'most important' word in a phrase, and the

word that gives the category to the phrase. For example, the phrase 'THE BLUE BOOK' is headed by the noun 'BOOK' and is thus a NP. Naturally, the definition of a head is not actually always this straightforward. For example, the head of the phrase 'THE FATHER'S CARS' is 'CARS', not 'FATHER', even though both words are nouns and the phrase is a NP. This is because the number of 'CARS' is plural, whereas the number of 'FATHER' is singular. However, the number of the entire phrase is plural, indicating that 'CARS' is the word that *passes up* the number feature, making 'CARS' the head of the phrase. More formally, a *head* can be defined as the word or construction that determines the type and feature values of the entire phrase. Feature values are generally passed to the phrase-level only from the head and from minor categories such as determiners. More details on heads can be found in (Bresnan, 2001).

In order to understand the following discussion, the most important concepts to remember are the following:

- **Pseudo-unification:** Pseudo-unification assigns the result of a unification to the left-hand side of the constraint equation. If the feature was not marked on the left-hand side, the feature value is merely passed from the right-hand side to the left-hand side.
- **Head:** A head can be defined as the word or construction that determines the type and feature values of the entire phrase. Feature values are passed to the phrase-level from the head.

As described above, constraints can refer to the x-side (SL), the y-side (TL), or both. They can be categorized as follows:

- **Value Constraints:**
  e.g. ((X1 NUM) = S)
  This constraint implies that the rule in which it occurs only applies if X1's number is singular, or if X1 does not mark for number. The latter can happen for example if a word is not recognized by the morphology module, or if it is ambiguous.
- **Agreement Constraints:**
  e.g. ((X2 NUM) = (X1 NUM))
  A rule with this constraint can only apply if the number values of X2 and X1 are the same, or if one or both of them do not mark for number.

The application of this constraint also means that the number value is now passed to X2. In other words, if X2 does not mark for number, but X1 does, the application of this constraint implies that X1's number value is passed to X2. This concept is especially important for cross-lingual constraints such as ((Y2 NUM) = (X1 NUM)), where X1's number constraint is passed to Y2, i.e. a feature is passed from the SL to the TL.

- **Head Passing Constraints:**
  e.g. (X0 = X2)
  The application of this constraint causes *all* feature values that are marked on X2 to be passed to X0. Constraints of this form are only used for heads of phrases in our system, where X2 is the head of a phrase and X0 stands for the phrase-level node. Then this rule passes all features from the head to the constituent level.

Head passing constraints are a subtype of agreement constraints that deserve separate treatment. They are necessary to determine features of constituents (rather than individual words), e.g. the number of an NP is determined by the number of the head. A head-passing constraint is used to ensure that the NP receives the number feature value. Due to their essential importance to any system that uses unification constraints, head passing constraints are *always* introduced both for the x-side and the y-side during Basic Constraint Learning (see Section 8.4) whenever Constraint Learning is done. In that section, we will discuss head passing constraints in greater detail and argue why they are important.

By contrast, in the following section on the taxonomy of constraints, we focus on value and agreement constraints. This is because value and agreement constraints should only be introduced as appropriate, whereas any rule with constraints is annotated with head passing constraints.

8.3 Taxonomy of Constraints

As mentioned in the previous section, two basic types of constraints are of interest to this discussion: value constraints and agreement constraints. Furthermore, a constraint can refer to the x-side, the y-side, or both. Constraints can moreover refer to individual words (raised to the part-of-speech level), to higher-level constituents, or to both. Other possible criteria of interest could be whether an agreement constraint constrains two heads, two non-heads, or a head and a non-head. These are only three examples of

parameters by which constraints can be distinguished. A few others will be described below.

The taxonomy that results from distinguishing constraints along these axes provides insight into what constraints may be useful for the given system: which constraints are learnable from text, and which ones are useful for translation? For those parameters that are of interest to our system, we propose a method for learning such constraints from text. The approach will encompass two stages: first, to detect cases when a constraint of a certain type may be needed, and second, to find and introduce the appropriate constraint. The learning approach is an extension of our earlier work (Probst et al., 2003).

In the following, we discuss a number of parameters along which unification constraints could vary. We give examples for each possible value, and explain those parameters that have not yet been discussed. We then argue which parameters are of interest to our system.

8.3.1 Constraint Parameter: Value or Agreement

Possible values: ***Value, Agreement.***

Examples for each possible value:

Value: ((X1 NUM) = P)
Agreement: ((X1 NUM) = (X3 NUM))

This parameter is of high interest to our system. Value constraints enforce that a component must be marked with a specific value. This can 1) limit the applicability of rules to certain contexts, or 2) limit the possible outputs. Agreement constraints enforce that two components are marked with the same feature value. This can also 1) limit the applicability of rules to certain cases, or 2) limit the possible outputs by enforcing that two TL components have the same value. In addition, however, there is another possibility: an agreement constraint can pass a feature from the SL to the TL, thereby also limiting the number of possible outputs.

8.3.2 Constraint Parameter: Level

Possible values: *POS, Constituent, POS/Constituent.*

Examples for each possible value:

POS:
NP::NP [DET ADJ N] → [N ADJ]
(... ((X3 NUM) = P) ...), also
NP::NP [DET ADJ N] → [N ADJ]
(... ((X3 NUM) = (X1 NUM))) ...)
Constituent:
S::S [NP V] → [V NP]
(... ((Y2 NUM) = P) ...), also
S::S [NP V] → [V NP]
(... ((Y2 NUM) = (X1 NUM)) ...)
POS/Constituent:
S::S [NP V] → [V NP]
(... ((X2 NUM) = (X1 NUM)) ...)

To clarify, constraints at the POS level constrain indices in the component sequences that are POS labels. In the first example, the constraint '((X3 NUM) = P)' constrains the third x-side index, i.e. the 'N'. By contrast, constraints at the constituent level constrain indices in the component sequences that are constituent labels. In the first constituent level example, '((Y2 NUM) = P)' constrains the 'NP' in the y-side component sequence.

Although it is not immediately obvious why this is so, this parameter is of interest to our system. In particular, y-side value constraints are different depending on whether they constrain a single word or an entire constituent, as will be discussed below.

As was noted above, words that remain lexicalized in the component sequence cannot be annotated with unification constraints (or alignments). For this reason, constraints can only constrain either POS indices or constituent indices.

8.3.3 Constraint Parameter: Language

Possible value: *x-side, y-side, xy.*

Examples for each possible value:

SL (x-side): ((X3 GEN) = F), also ((X2 GEN) = (X5 GEN))
TL (y-side): ((Y3 NUM) = P), also ((Y2 NUM) = (Y3 NUM))
SL→TL (xy): ((Y3 NUM) = (X5 NUM))

This parameter is of high interest to our system. x-side constraints limit the applicability of the rule, xy-constraints determine what feature values are passed to the target language, and y-side constraints limit the possible outputs of a rule.

8.3.4 Constraint Parameter: Constrains Head

Possible values: *Head, Non-Head, Head/Non-Head.*

Examples for each possible value:

Head:
NP::NP [N ADJ] → [DET ADJ N]
(... ((Y3 NUM) = P) ...), also
NP::NP [N ADJ] → [DET ADJ N]
(... ((Y3 NUM) = (X1 NUM))) ...)
Non-Head:
NP::NP [N ADJ] → [DET ADJ N]
(... ((Y2 GEN) = F) ...), also
NP::NP [N ADJ] → [DET ADJ N]
(... ((Y1 GEN) = (Y2 GEN))) ...)
Head/Non-Head:
NP::NP [N ADJ] → [DET ADJ N]
(... ((Y3 GEN) = (Y2 GEN)) ...)

These constraints are discussed below in the context of agreement constraints. In particular, the agreement constraints that will be introduced will be exactly those constraints that will pass features to heads or from the SL to the TL. In the case of x-side and y-side constraints, features

are passed from non-heads to heads. In the cross-lingual case, features are passed between aligned indices (either two heads or two non-heads).[1]

The following three parameters (Depth, Enforce Existing Value, Multiple Values) are not addressed in our system, but are discussed here because they are frequently mentioned in the context of unification-based formalisms.

8.3.5 Constraint Parameter: Depth

Possible values: ***1, 2,*** ...

Examples for each possible value:

1: ((X4 DEF) = +)
2: ((X3 SUBJ NUM) = P)
...

Depth of constraints has not yet been defined in this thesis. The depth of a constraint is the length of the 'path' of features that are marked. This is best illustrated with an example: '((X3 SUBJ NUM) = P)' does *not* mean that X3 should be marked for plural, but rather that X3's *subject's* number value should be plural. The expression 'X3's subject's number' is the path. Its length is the depth of the constraint. In our system, we restrict ourselves to constraints of depth 1, i.e. simple constraints of feature-value pairs. The reason is that we do not learn function assignments, such as subject, object, etc. for the components, because such assignments are not available to us and cannot be obtained only from the parse in a reliable way.

8.3.6 Constraint Parameter: Enforce Existing Value

Possible values: ***'=', '=c'.***

Examples for each possible value:

=: ((X4 DEF) = +)
=c: ((X4 DEF) =c +) This will only succeed if X4 is already marked for

[1]The reader is reminded that there are also head constraints, where all features are passed from the head of a phrase to the X0 or Y0 level. These constraints are automatically introduced and do not fall under this discussion.

definiteness before the constraint is applied.

Unification formalisms generally allow for a special type of unification that is not handled in our system, and has not yet been discussed. This is the concept of a different type of unification, namely '=c'. =c is used instead of =, e.g., in a constraint such as '((X1 NUM) =c (X2 NUM))'. This differs slightly from '((X1 NUM) = (X2 NUM))' in semantics: The unification '((X1 NUM) =c (X2 NUM))' will succeed only if 1) X1 and X2 are marked for number before the unification equation is evaluated, and 2) X1 and X2 have the same number value. By contrast, the unification of the constraint '((X1 NUM) = (X2 NUM))' will succeed if 1) either or both X1 and X2 are *not* marked for number, *or* 2) both are marked with the same number value. In our system, this distinction is not handled, as it is intricately associated with the strength of the morphology modules. Since the morphology modules are not complete and sometimes give wrong information, it is not desirable to enforce that a specific value *must* be marked. It could happen that a specific feature is simply not returned by the morphology module, which would prevent an entire rule from applying. For this reason, we conservatively refrain from introducing =c constraints.

8.3.7 Constraint Parameter: Multiple Values

Possible value: *AND, OR, Single Value.*

Examples for each possible value:

and: (∗AND∗ (((X1 NUM) = P) ((X2 DEF) = +)))
or: ((X1 PER) = (∗OR∗ 1 2))
single value: ((X1 NUM) = P)

Multiple values have also not yet been discussed. In the above examples, an 'OR' implies that the unification will succeed if one or more of the ground values are marked on X1, i.e. it will succeed if X1's person value is either 1 or 2. A set of unification constraints that is surrounded by AND will only succeed if each of the unifications in the set succeeds, otherwise it fails.

In our system, OR can be used just like any other ground value; it makes no difference to the rule learning whether it is faced with a single or a composite ground value. AND is in fact implicit in our rule applications: a rule can only apply if all unification constraints in the rule succeed. For this

reason, neither OR nor AND will receive special attention in our discussion of unification constraints.

8.3.8 Subtypes of Constraints

The above discussion leaves three parameters that are of interest to our system: value or agreement, level, and language. Forming the cross-product between all possible values of these three parameters results in 18 possible combinations, as can be seen in Table 8.1. The table indicates that, for example, a value constraint that constrains a constituent on the x-side (SL) may have to be treated differently than a value constraint that constrains a constituent on the y-side (TL). In other words, the three parameters are not independent.

Val/Agr	**Language**	**Level**	**Type**
value	x	POS	
value	x	const	**x-side value constraints**
value	x	POS/const	cannot exist
value	y	POS	**y-side POS value constraints**
value	y	const	**y-side const value constraints**
value	y	POS/const	
value	xy	POS	
value	xy	const	cannot exist
value	xy	POS/const	
agreement	x	POS	
agreement	x	const	**x-side agreement constraints**
agreement	x	POS/const	
agreement	y	POS	
agreement	y	const	**y-side agreement constraints**
agreement	y	POS/const	
agreement	xy	POS	
agreement	xy	const	**xy agreement constraints**
agreement	xy	POS/const	

Table 8.1: List of relevant constraint types.

Learning is facilitated by the fact that some of these categories are not of interest to our system, and because some categories can be collapsed. As can be seen from the table above, some combinations, such as 'value,x,POS/const' cannot exist (because a value constraint constrains

only one index, and POS/const requires two constrained indices). Subtypes that can be collapsed are shown above with the same group name, e.g., x-side value constraints. We will now give examples of all relevant constraint types. In subsequent sections, it will become clear why some of the subtypes can be collapsed into groups. This has to do with their impact on translation.

Examples of x-side value constraints:

((X1 NUM) = P)

Examples of y-side POS value constraints:

NP::NP [N ADJ] → [DET ADJ N]
(... ((Y3 NUM) = P) ...)

Examples of y-side constituent value constraints:

S::S [NP V] → [V NP]
(... ((Y2 NUM) = P) ...)

Examples of x-side agreement constraints:

S::S [NP V] → [V NP]
(... ((X2 NUM) = (X1 NUM)) ...)

Examples of y-side agreement constraints:

S::S [NP V] → [V NP]
(... ((Y1 NUM) = (Y2 NUM)) ...)

Examples of xy agreement constraints:

S::S [Det N V] → [V N]
(... ((Y2 NUM) = (X2 NUM)) ...)

8.4 Learning Basic Constraints

After discussing the types of constraints that could exist and those that are meaningful to our task, we now present how these constraints are learned in our system.

This section discusses a set of constraints that are introduced in all rules as a starting point for Constraint Learning. With the exception of head passing constraints, only value constraints are introduced in this phase. The basic constraints essentially gather as much information as can be obtained from the morphology modules and introduce a large set of constraints. The subsequent Constraint Learning phases then use the basic constraints as input, generalize them to agreement constraints, and subject them to an appropriateness test to see if they should be retained as value constraints in the final rules.

It is important to note here that if a rule was produced by more than one training example, then during Constraint Learning we distinguish the rules for each training example. For example, if we learn only context-free rules, we collapse rules that are the same but were produced by two training examples. During Constraint Learning, we would consider them two separate rules. This is because each set of words (i.e. training example) can give rise to a unique set of basic constraints. If the grammar with constraints contains duplicates at the end of learning, the duplicate rules are once more collapsed.

The Basic Constraint Learning phase relies on two sources of information: the knowledge of feature values for specific words, and the knowledge of what word is the head of a given phrase in both languages. We will first describe how these two pieces of information are obtained, and will subsequently describe how they are used to form basic constraints.

First, features for specific words can be determined from a morphology module. Each rule is learned from a bilingual training example. Each of the words in the training example is passed through the morphology modules for the respective language. The modules return a set of features for each word. Sometimes, disambiguation is necessary when more than one analysis is found for a given word. Generally, however, there is only one possible analysis per part of speech. During Seed Generation, we have already performed POS disambiguation. For the TL, the parts of speech can be obtained from the parse that is given for each bilingual training example. For the SL, we use the alignments as well as the morphology module to assign parts of speech to individual words. For more details, the reader is referred back to chapter 6. During Basic Constraint learning, the same POS

assignments are used for the individual words. We then extract all relevant features for the disambiguated part of speech assignments.

Second, we can determine with acceptable accuracy what the head of a phrase is. For this, we use a version of Michael Collins's head Table (Collins, 1996) that was adapted to our tagset. The head table specifies how, for English, heads can be found in different types of phrases. For instance, the head of an NP is generally the rightmost noun. The head table is heuristic, but yields good results and is routinely used in the statistical parsing community.

The SL head cannot be found as easily. In the absence of any other information such as a head table for SL, we assume the SL head to be the word that is aligned to the TL head. Some complications arise when the TL head is not aligned to exactly one SL head. In the case of multiple alignments, we prefer the SL word that is of the same POS as the TL word. If the TL head is unaligned, no SL head can be found. Similarly to the TL head table, the mechanism for finding the SL head is heuristic, but yields good results. A future research direction could be to explore head finding in greater detail, however this is considered to be outside the scope of the thesis.

With feature values on specific words and head information now available, we can construct the set of basic constraints. For components that cover only one word, the features for this word are introduced as value constraints on those indices. For example, if a rule

```
NP::NP ... -> [DET ADJ N]
```

was obtained from a training example whose English side is 'THE BLUE CAR', then the rule would have a constraint of the form ((Y3 NUM) = S), because the English morphology module returns a number value of singular for the noun 'CAR'. In other words, for words in the training example the are captured as PTs (i.e. POS labels) in the component sequences, we can simply introduce value constraints for those indices.

This is however not enough. As was discussed above, a constituent can only carry a feature if this feature was passed up from one of the words in the component. Consider the rule

```
NP::NP ... -> [DET N "'s" N]
```

derived from the example sentence 'THE FATHER'S CARS'. Y0 should be marked for definiteness and with a number value of plural, for reasons discussed below. For example, this rule should not combine with a rule that

should apply only to singular NPs. This means that we must introduce constraints for the constituent level.

Feature values are 'passed up' by marking certain feature values as value constraints at the constituent level, i.e. the phrase level. For example, Figure 8.1 shows a partial rule that is marked up with basic constraints. Some value constraints are marked on indices referring to components in the component sequence. Additional constraints however are inserted for the constituent (phrase) level Y0. These feature values were derived from the words in the bilingual training example, and were 'passed up' from them to the constituent level. Generally it is not obvious what feature values should be passed up to the component level, and from what words they should be passed up. As was said before, feature values are generally passed up from the head and from minor categories to the constituent level. In our approach, we pass up features from all words, because the subsequent learning phases only retain those constraints that are appropriate for the given rule. For example, the phrase 'THE CHILDREN'S ROOM' is considered definite, although definiteness is marked only on the determiner 'THE', not on the head 'ROOM', as can be seen in the rule in Figure 8.1 below. Complications arise when there is a contradiction between a feature value from a head and a feature value from a non-head. In such a case, the feature value from the head will prevail, and the non-head's feature value will not be marked at the constituent level. For example, the number value of the head 'ROOM' overrides the number value of the non-head 'CHILDREN', as follows:

```
{NP,1}
;;SL: ...
;;TL: THE CHILDREN 'S ROOM
NP::NP ... -> [DET N "'s" N]
(
...
((Y1 DEF) = +)
((Y2 NUM) = P)
((Y4 NUM) = S)
((Y0 DEF) = +)
((Y0 NUM) = S) ;overrides NUM value from 'CHILDREN',
        ;because 'room' is tagged as the head
)
```

Figure 8.1: Sample rule with basic constraints.

This approach can sometimes cause problems, since from a linguistic perspective not necessarily all constraints should be passed up. While this leaves opportunities for future research, it is a reasonable approach in the current system for the reason cited earlier: subsequent Constraint Learning phases eliminate those constraints that are not necessary for translation. This will be described in greater detail below.

One may wonder why it is even important to mark feature values on the phrase (constituent) level. The following example should illuminate this point. Suppose we have a rule such as the following:

```
{PP,1}
PP::PP [PREP NP] -> [PREP NP]
(
...
((Y2 DEF) = -)
)
```

The constraint in this rule essentially enforces that the y-side NP in the translation be indefinite. Suppose the run-time system would try to use this rule in conjunction with the previous rule NP,1. A human inspector would realize that the combination of the two rules should not succeed, because the y-side NP in the NP,1 is definite, not indefinite. This is in fact what happens, and the two rules do not apply in combination. But why? The reason is that in the NP rule, definiteness is marked at the Y0 level. If the Y0 NP is used in a different rule, such as PP,1 above, then all features marked on Y0 must unify with all features for Y2 (the NP slot). If they do not, as in this example, then the two rules cannot combine. For this reason, it is important to pass relevant features to the X0 and Y0 level. As this is not a thesis about unification theory, we must refer the reader to sources such as (Bresnan, 2001) for further details.

The question remains what should be done about cases where sequences of words were generalized to constituents during Compositionality Learning? In other words, it must be determined what should be done with rules whose component sequences contain constituent labels such as the 'PP' in the following rule:

```
NP::NP [DET ADJ N PP] -> ...
```

In such cases, Basic Constraint learning must be applied recursively. Consider the following example:

```
S::S ... -> [NP V]
```

How can we accomplish that Y2 is marked for number? This is done with the same mechanism as for non-compositional components. Value constraints for components are marked up with value constraints in the same way as we previously passed up features to the X0 and Y0 level.

As mentioned earlier, during Basic Constraint learning we also introduce a set of head passing constraints, i.e. constraints that pass all features from one component, namely the head, to the X0 or Y0 level. Since we are generally able to determine the head of a phrase as described above, we introduce such head passing constraints for both the x- and the y-side. These basic agreement constraints are always retained in the rule.

Figure 8.2 summarizes Basic Constraint Learning in pseudocode.

8.5 Learning Agreement Constraints

(x-side agreement constraints, y-side agreement constraints, xy agreement constraints)

In the previous section, we described the first phase of Constraint Learning: the introduction of basic constraints from morphology, and their 'passing up' from the word to the phrase (constituent) level. In this and the subsequent sections, we turn our attention to the constraints types listed in Table 8.1. The first type of constraints we will address are agreement constraints. Agreement constraints are generalizations of value constraints, because they do not specify a ground value, but rather only enforce that two components are marked with the *same* value, whatever that constraint may be. Agreement constraints can be obtained from pairs of value constraints. For this reason, Agreement Constraint Learning utilizes the value constraints introduced during Basic Constraint Learning, and produces agreement constraints from them wherever possible. The subsequent learning phase will then eliminate those value constraints that are not useful for translation.

There are three different kinds of agreement constraints: agreement constraints may pertain only to one language (either the SL or the TL), or they may be cross-lingual constraints. More specifically:

1. **SL constraints (x-side agreement constraints):** Such constraints limit the applicability of the rules and thus parsing ambiguity. This category also includes constraints that pass features up to the constituent level, such as ((Y0 NUM) = (Y1 NUM)).

2. **TL constraints (y-side agreement constraints):** Such constraints limit the output of the rules and thus generation ambiguity. As in the

Learning Basic Constraints

For each learned rule r_i and the corresponding training example tr_i with words $w_{1,SL}$ ... $w_{m,SL}$ and $w_{1,TL}$ ... $w_{n,TL}$ and component sequences $comp_{1,SL} \ldots comp_{m',SL}$ for the SL and $comp_{1,TL} \ldots comp_{n',TL}$ for the TL

- Find head of SL sentence: $head_{SL}$ = FindHead($w_{1,SL} \ldots w_{m,SL}$, SL)
- Get index $ind_{head_{SL}}$ in SL component sequence corresponding to $head_{SL}$
- Insert into rule agreement constraint (X0 = X$ind_{head_{SL}}$)
- For each $comp_{i,SL} \in comp_{1,SL} \ldots comp_{m',SL}$
  - Get index $ind_{comp_{i,SL}}$ corresponding to $comp_{i,SL}$
  - Get all words $w_{a,SL}$ ... $w_{b,SL}$ covered by $comp_{i,SL}$
  - Find head $head_{comp_{i,SL}}$ of $comp_{i,SL}$: $head_{comp_{i,SL}}$ = FindHead($w_{a,SL}$ ... $w_{b,SL}$,SL)
  - For each word $w_{a,SL}$ ... $w_{b,SL}$, get all feature-value pairs from SL morphology
  - Insert value constraints for index $ind_{comp_{i,SL}}$ for all feature-value pairs
  - In case of conflicts, the feature value marked on $head_{comp_{i,SL}}$ prevails
- Repeat everything for TL

FindHead(WordSeq, Language)

- Extract POS sequence for WordSeq, derived during Seed Generation
- If Language = TL, use head table[a] to find head
- Else if Language = SL, project head from TL using user-specified word alignments
- Return head

[a] The head table can be found at (Collins, 1996).

Figure 8.2: Pseudocode for Basic Constraint Learning.

previous case, this category includes constraints that pass features up to the constituent level, in this case X0.

3. **SL→TL constraints (xy agreement constraints):** With SL→TL constraints, feature values are passed from the source language to the target language; by this process, they limit the possible output of rules, more specifically by limiting transfer ambiguity.

Agreement constraints are learned from the value constraints produced during Basic Constraint learning. This is achieved in three distinct phases. The first two phases will focus on intra-lingual constraints, i.e. SL-only constraints and TL-only constraints. Since one language does not influence what agrees with what else in the other language (e.g., Hebrew has no influence on subject-verb agreement in English), these constraints can be learned separately. This is an important observation, because separation into smaller tasks can make learning easier and more reliable. The third phase will focus on those feature values that should be passed from the SL to the TL. Finally, the fourth phase will determine which value constraints should be kept and which ones have become obsolete because of the agreement constraints.

As the three phases of Agreement Constraint Learning focus on agreements rather than specific values, the values are deemphasized. In other words, it is more important whether two components *agree* in a value than what this specific value is. More specifically, we are interested in how often we observe the *same* value and how often we observe a *different* value. The following discussion treats the case of cross-lingual constraints, i.e. feature values being passed from SL to TL. The intra-language case is defined equivalently, as we will see.

Suppose a feature can take values $v_1, v_2, \ldots, v_k$. For the purpose of discussion (so that we can show a concrete contingency table), assume that $k = 3$. The approach works equivalently for other values of k. Denote $c_{i,SL}$ as the source language component (constituent or POS) that is constrained, and denote $c_{j,TL}$ as the target language component that is aligned to $c_{i,SL}$ and that is constrained on the same feature. Then we can construct a contingency table as follows:

	$c_{i,SL} = v_1$	$c_{i,SL} = v_2$	$c_{i,SL} = v_3$	
$c_{j,TL} = v_1$	$m_{1,1}$	$m_{2,1}$	$m_{3,1}$	$m_{x,1}$
$c_{j,TL} = v_2$	$m_{1,2}$	$m_{2,2}$	$m_{3,2}$	$m_{x,2}$
$c_{j,TL} = v_3$	$m_{1,3}$	$m_{2,3}$	$m_{3,3}$	$m_{x,3}$
	$m_{1,y}$	$m_{2,y}$	$m_{3,y}$	N

The number of corresponding constraints with the same value is $m_{same} = m_{1,1} + m_{2,2} + m_{3,3}$. The number of constraints with different values is $m_{different} = N - m_{same}$. Agreement Constraint Learning looks at whether m_{same} is considerably larger than $m_{different}$, i.e. whether $m_{same} >> m_{different}$. Whether this is the case is determined either by a heuristic or a statistical significance test, as will be described below. If m_{same} is in fact much larger than $m_{different}$, then it can be concluded that most of the time the value of this constraint is the same, and that an agreement constraint should be introduced instead of the less general value constraints.

During Agreement Constraint Learning, no value constraints are eliminated - this phase merely introduces additional constraints. Elimination of irrelevant constraints is not done until the last learning phase (section 8.6).

We begin the process by considering several pieces of information:

1. First, we create a list of all possible index pairs that should be considered for agreement constraints.

   - SL-only constraints: collect a list of all **head/non-head pairs in the same constituent** that occur with the same feature (not necessarily same value). For example, we will consider the following for an agreement constraint: DET and N in a NP where the DET is a dependent of the N and where DET and N mark the same feature, e.g., NUM, not necessarily with the same value. We then scan through all the rules and find all those rules with a DET and a N in a NP where the DET is a dependent of the N.
   - TL-only constraints: same as SL only constraints above.
   - SL→TL constraints: consider all situations where two **aligned components mark the same feature** (not necessarily with the same value).

2. After creating a set of rules for each situation, we then collect the counts of how many times the feature value was the same, denoted by E_{same}, and how many times the feature was different, $E_{different}$. Note that we now have to keep track of how many times one of the components did not mark the feature at all. We exclude from the hypothesis or heuristic test those rules that do not mark the feature on both components. If the hypothesis test leads to the conclusion that an agreement constraint should be introduced, then all rules of this situation will receive an agreement constraint, even though the original rule did not mark the feature on both components.

The question remains how to weigh the positive and negative evidence in order to get a reasonable estimate of how confident we should be in introducing an agreement constraint. Note that some of these counts will be very low when the learning corpus is small. The simplest way is to introduce an agreement constraint if we have more positive than negative evidence:

Introduce an agreement constraint if $E_{same} \geq E_{different}$.

This is a heuristic test, but a good solution when little training data is given. Alternatively, we can use a likelihood ratio test to determine how likely it is that the values are dependent on each other (force each other to be the same), and how likely it is that they are distributed independently.

The likelihood ratio test is done by first proposing two hypotheses of what gave rise to the data:

- **Hypothesis 0 (Θ_0):** The values are independently distributed.
- **Hypothesis 1 (Θ_1):** The values are not independently distributed.

We then compute the likelihood of the data given each of the two hypotheses and take the ratio between the two. Alternatively, to avoid numeric underflow, we can take the log-likelihoods and take the difference between them. ll_{Θ_0} is the log-likelihood of the data under the null hypothesis. It can be computed as follows:

$$ll_{\Theta_0} = \sum_{i=1}^{n} log(p_{v_{x_{i1}}} * p_{v_{x_{i2}}})$$

Here, the two indices in the compared pair (i.e. the head/non-head pair in the intralingual case and the aligned indices in the interlingual case) are denoted by x_{i1} and x_{i2}. $v_{x_{i1}}$ is the value that index x_{i1} is marked with, and equivalently, $v_{x_{i2}}$ is the value that index x_{i2} is marked with. $p_{v_{x_{i1}}}$ is the probability of the occurrence of this value. How this probability can be estimated will be discussed below.

Under the alternative hypothesis, we can determine the likelihood as follows:

$$ll_{\Theta_1} = \sum_{i=1}^{n} log(p_{v_{x_{i1}}} * ind)$$

where ind is 1 if $v_{x_{i1}} = v_{x_{i2}}$ and 0 otherwise. In other words, under the alternative hypothesis we merely assign probabilities based on whether the

value in the two compared indices is the same or whether it is different. If it is different, then no 'credit' is given to this compared pair.

The difference ratio between those two, $ll_{\Theta_0} - ll_{\Theta_1}$ (difference rather than ratio because of the logs) is then an indicator of how much more likely the null hypothesis is than the alternative hypothesis. We use a χ^2 table to find whether we should reject the null hypothesis at a level of 0.1 confidence.

Let us now turn to the discussion of how the probabilities of the occurrence of specific values can be estimated. Again, x_{i1} and x_{i2} are the compared indices that are considered for generalization to an agreement value. The value for each index, $v_{x_{i1}}$ and $v_{x_{i2}}$, is modeled as drawn from a multinomial distribution. The probabilities for each value can be estimated from the data as a whole using the Maximum Likelihood Estimator: this will is the proportion of the time we see the given feature with a value v_i. Then:

$$p_{v_i} = \frac{c_{v_i}}{\sum_{j=1}^{k} c_{v_j}}$$

where c_{v_i} is the number of times the value v_i (e.g., 1st) was encountered for the given feature (e.g., PERS), and k is the number of possible values for the feature (e.g., 1st, 2nd, 3rd).

If there is strong evidence that the two compared indices will generally agree with each other on this feature, then we introduce an agreement constraint *without eliminating any of the value constraints*. As the transfer engine uses pseudo-unification, it matters which index is on the left-hand side of the agreement constraint: in the intralingual case (SL-only and TL-only constraints), it should be the head index. This implies that the value from the dependent is passed up to the head. In the interlingual case, the left-hand index in the constraint should be the y-side index, so that the feature can be passed from the SL to the TL.

In our experiments, we apply the log-likelihood ratio test if there considerable data for a specific case (n $\geq$ 10). Otherwise, we revert to the much simpler heuristic solution where an agreement constraint is introduced if there is more evidence for than against it, i.e. if $E_{same} \geq E_{different}$.

8.5.1 SL Agreement Constraints

(x-side agreement constraints)

In this phase, we consider only the SL side of the learned rules, independently of the TL. As was discussed above, in the case of x-side agreement

constraints, we collect a list of all head/non-head pairs in the same constituent that occur with the same feature (not necessarily same value). For example, we will consider the following for an agreement constraint: DET and N in a NP where the DET is a dependent of the N and in at least one training example DET and N mark the same feature, e.g., NUM, not necessarily with the same value. We then scan through all the rules and find all those rules that exhibit the property DET and N in a NP where the DET is a dependent of the N.

Agreement constraints are then introduced as described in the previous section. Pseudocode for the learning of SL agreement constraints can be found in Figure 8.3.

8.5.2 TL Agreement Constraints

(y-side agreement constraints)

Learning TL constraints proceeds in the same fashion as learning SL agreement constraints. Equivalently to the previous section, the other language, in this case the SL, is ignored. Again, we collect evidence from all head/non-head pairs to determine what agreement constraints should be introduced. No value constraints are eliminated; this phase only introduces new constraints if appropriate. Pseudocode for the learning of TL agreement constraints can be found in Figure 8.4.

8.5.3 SL→TL Agreement Constraints

(xy agreement constraints)

This is defined similarly to the previous phases, except that in this case we consider cross-lingual constraints. Evidence will be collected from aligned pairs of indices for a specific SL type, a specific TL type (that is aligned to the SL index in question), and a specific feature.

As was mentioned above, a necessary modification to the intralingual algorithm is that in the interlingual case, the y-side index (rather than the head as previously) is always the left-hand side index of the constraint equation. The reason is that the run-time system uses pseudo-unification, and that in pseudo-unification, the result of the unification is stored only in the left-hand side index. By making the y-side index the left-hand side of the agreement constraint, the feature value can be passed from the SL to the TL. Pseudocode for the learning of SL-TL agreement constraints can be found in Figure 8.5.

Advanced Constraints: Learning SL agreement constraints

Collect from all rules all sets S of *head-nonhead* SL component pairs that occur with the same feature f_i on the same types:

$S = \{pair_i = \langle comp_{i,1,SL}, comp_{i,2,SL}\rangle |$
$\forall i, type_{comp_{i,1,SL}} = type_1, type_{compi,2,SL} = type_2,$
$\exists$ feature f_j s.t. f_j marked on
$comp_{i,1,SL}$,$comp_{i,2,SL}$
with values $v_{f_j,comp_{i,1,SL}}$,$v_{f_j,comp_{i,2,SL}}$ $\}$

For each such set S

If |S| < 10, perform heuristic test:

Get
c_{same}=$|\{pair_i = \langle comp_{i,1,SL}, comp_{i,2,SL}\rangle \in S|$
$v_{f_j,comp_{1,SL}} = v_{f_j,comp_{2,SL}}\}|$ and
c_{diff}=$|\{pair_i = \langle comp_{i,1,SL}, comp_{i,2,SL}\rangle \in S|$
$v_{f_j,comp_{1,SL}} \neq v_{f_j,comp_{2,SL}}\}|$
If $c_{same} \geq c_{diff}$, test *passes*
Else test *fails*

Else perform log-likelihood ratio test:

Compute likelihood for Θ_0 ($v_{f_j,comp_{1,SL}}$ and $v_{f_j,comp_{2,SL}}$ are independently distributed) using estimation described in 8.5
Compute likelihood for Θ_1 ($v_{f_j,comp_{1,SL}}$ and $v_{f_j,comp_{2,SL}}$ are *not* independently distributed) using estimation described in 8.5
If Θ_0 rejected, test *passes*
Else test *fails*

If heuristic or likelihood ratio test *passed*

Introduce agreement constraint for f_j for all rules $\in S$ where $v_{f_j,comp_{1,SL}} = v_{f_j,comp_{2,SL}}$

Figure 8.3: Pseudocode for Advanced Constraint Learning: learning SL agreement constraints.

Advanced Constraints: Learning TL agreement constraints

Collect from all rules all sets S of *head-nonhead* TL component pairs that occur with the same feature f_i on the same types:

$S = \{pair_i = \langle comp_{i,1,TL}, comp_{i,2,TL}\rangle |$
$\forall i, type_{comp_{i,1,TL}} = type_1, type_{compi,2,TL} = type_2,$
$\exists$ feature f_j s.t. f_j marked on
$comp_{i,1,TL}$,$comp_{i,2,TL}$
with values $v_{f_j,comp_{i,1,TL}}$,$v_{f_j,comp_{i,2,TL}}$ }

For each such set S

If |S| < 10, perform heuristic test:

Get
c_{same}=$|\{pair_i = \langle comp_{i,1,TL}, comp_{i,2,TL}\rangle \in S|$
$v_{f_j,comp_{1,TL}} = v_{f_j,comp_{2,TL}}\}|$ and
c_{diff}=$|\{pair_i = \langle comp_{i,1,TL}, comp_{i,2,TL}\rangle \in S|$
$v_{f_j,comp_{1,TL}} \neq v_{f_j,comp_{2,TL}}\}|$
If $c_{same} \geq c_{diff}$, test *passes*
Else test *fails*

Else perform log-likelihood ratio test:

Compute likelihood for Θ_0 ($v_{f_j,comp_{1,TL}}$ and $v_{f_j,comp_{2,TL}}$ are independently distributed) using estimation described in 8.5
Compute likelihood for Θ_1 ($v_{f_j,comp_{1,SL}}$ and $v_{f_j,comp_{2,SL}}$ are *not* independently distributed) using estimation described in 8.5
If Θ_0 rejected, test *passes*
Else test *fails*

If heuristic or likelihood ratio test *passed*

Introduce agreement constraint for f_j for all rules $\in S$ where $v_{f_j,comp_{1,TL}} = v_{f_j,comp_{2,TL}}$

Figure 8.4: Pseudocode for Advanced Constraint Learning: learning TL agreement constraints.

Advanced Constraints: Learning SL-TL agreement constraints

Collect from all rules all sets S of *SL-TL* component pairs that occur with the same feature f_i on the same types:

$S = \{pair_i = \langle comp_{i,SL}, comp_{i,TL}\rangle |$
$\forall i, type_{comp_{i,SL}} = type_{SL}, type_{compi,TL} = type_{TL},$
$\exists$ feature f_j s.t. f_j marked on $comp_{i,SL}$,$comp_{i,TL}$
with values $v_{f_j,comp_{i,SL}}$,$v_{f_j,comp_{i,TL}}$ $\}$

For each such set S

If |S| < 10, perform heuristic test:

Get
c_{same}=$|\{pair_i = \langle comp_{i,SL}, comp_{i,TL}\rangle \in S|$
$v_{f_j,comp_{SL}} = v_{f_j,comp_{TL}}\}|$ and
c_{diff}=$|\{pair_i = \langle comp_{i,SL}, comp_{i,TL}\rangle \in S|$
$v_{f_j,comp_{SL}} \neq v_{f_j,comp_{TL}}\}|$
If $c_{same} \geq c_{diff}$, test *passes*
Else test *fails*

Else perform log-likelihood ratio test:

Compute likelihood for Θ_0 ($v_{f_j,comp_{SL}}$ and $v_{f_j,comp_{TL}}$ are independently distributed) using estimation described in 8.5
Compute likelihood for Θ_1 ($v_{f_j,comp_{SL}}$ and $v_{f_j,comp_{TL}}$ are *not* independently distributed) using estimation described in 8.5
If Θ_0 rejected, test *passes*
Else test *fails*

If heuristic or likelihood ratio test *passed*

Introduce agreement constraint for f_j for all rules $\in S$ where $v_{f_j,comp_{SL}} = v_{f_j,comp_{TL}}$

Figure 8.5: Pseudocode for Advanced Constraint Learning: learning SL-TL agreement constraints.

8.6 Value Constraints Revisited

(x-side value constraints, y-side POS value constraints, y-side constituent value constraints)

x-side value constraints

After introducing basic constraints using the morphology and principles to pass features to the constituent level, we have examined which value constraints should be generalized to agreement constraints. During that latter phase, we did not eliminate any existing value constraints. We delayed the discussion and treatment of value constraints until this time, because it was necessary that the Agreement Constraint Learning phase had all value constraints available. Only in this phase can we now eliminate those value constraints that are not of interest for translation.

First, we will discuss the different types of value constraints, and how they can be of interest for translation, and then we will describe how an automatic decision is made about their usefulness.

x-side value constraints limit the applicability of a rule. At run-time, they only succeed if the value constraint unifies with the morphological analysis or the dictionary entry of the index in question, or else of the value that was passed up to the index by one or more lower-level rules. Limiting applicability can be called for under three circumstances: first, a rule should only apply if the value constraint is met, and a *different* value must be marked on the TL side. This cannot be expressed as an agreement constraint, as agreement constraints enforce the same, not a different value. For instance, in a situation where a certain SL NP must be in the singular, but the corresponding NP in the TL must be plural. This case will simply be handled as follows: if we detect a case where two corresponding indices mark for the same feature, but with different values, we retain both value constraints.

The second applicability limitation occurs when the corresponding TL index should carry the same value. This situation is handled by agreement constraints, so it is not of interest to the present discussion.

Finally, there are cases where a value constraint determines the *structure* of the TL translation. This can occur if a given SL context-free structure can translate into at least two different TL context-free structures, and there is a specific value that can determine which TL structure must be produced. The following example illustrates this problem. Here, the number value determines the structure of the TL output.

```
;;SL: ILD @WB
;;TL: A GOOD BOY
NP::NP [N ADJ] -> ["A" ADJ N]
(
...
((X1 NUM) = S)
((X2 NUM) = S)
((Y3 NUM) = S)
...
)

;;SL: ILDIM @WBIM
;;TL: GOOD BOYS
NP::NP [N ADJ] -> [ADJ N]
(
...
((X1 NUM) = P)
((X2 NUM) = P)
((Y2 NUM) = P)
...
)
```

In this section, we will examine such cases where an SL structure transfers into different TL structures: can the value of a constraint determine what structure should be produced in the TL?

To accomplish this, we begin by grouping the learned rules such than in each group the type and component sequence of the SL is constant. Denote this by the SL context-free part of the rule, CF_{SL_i}. In other words, we form all groups where CF_{SL_i} is constant. In all such groups, we then look for cases where the CF_{TL_i} is *different*, and check whether a specific SL feature value can distinguish between the different CF_{TL_i} structures.

In the above example, the value of the number feature determines the structure of the translation. If such a case is detected, and a feature is found to influence what type of structure is produced, its ground value is retained. In the above example, both rules retain the number constraint with the specific value (SG and PL, respectively). Otherwise, the value constraints are eliminated, and only the agreement constraints are retained.

After grouping the learned rules by their SL context-free parts, we examine how a given feature correlates with a specific TL component sequence:

Let CF_{TL} be the TL component sequence, let CF_{SL} be the SL compo-

nent sequence, and let v_i be a specific value associated with a specific feature (e.g., NUM value SG). Then define $P(CF_{TL}|v_i)$ only over those cases where CF_{SL} is constant.

$$P(CF_{TL}|v_i) = \frac{P(CF_{TL}, v_i)}{P(v_i)}.$$

This can be estimated by

$$\frac{\frac{c_{CF_{TL},v_i}}{c_{CF_{SL}}}}{\frac{c_{v_i}}{c_{CF_{SL}}}} = \frac{c_{CF_{TL},v_i}}{c_{v_i}}.$$

If this value is greater than 0.5, then there is good evidence that this particular ground value should be retained.

Similarly to Agreement Constraint Learning, we must have a fallback heuristic method if there is not enough training data to get a reliable estimate as described above. For this reason, if we have less than 10 training examples for a potential pair of value constraints, we only retain the value constraints if the training data is completely separable by them. This means, if we can retain value constraints that deterministically determine the structure of the TL translation without any mistakes on the training data, we do so.

To summarize, we have discussed three types of x-side value constraints, and how they are handled:

1. The value of the corresponding TL index is *different*. In such cases, the x-side value constraints are retained.

2. The value of the corresponding TL index is the *same*. In such cases, the x-side value constraint is not retained, but this case is handled by agreement constraints.

3. The value the x-side value constraints determines the component sequence, i.e. structure, of the TL. In this case, the value is retained if a different value would produce a different TL structure, and the ground values distinguish between the two TL structures.

y-side POS value constraints

y-side POS value constraints are of little interest to our system: these kinds of value constraints on the TL side constrain only one word. In our system setup, features on single words are passed in the lexicon, but the TL word

translations are not marked for those features. This means that y-side POS value constraints would have no effect on the resulting translation, because they would not be subjected to unification with any feature values passed from the lexicon or morphology. In other words, the constraints would always succeed. As was said above, we describe here how the taxonomy of constraints is applied to our system. In other systems, y-side POS value constraints may well play a role.

y-side constituent value constraints

y-side constituent value constraints are constraints that limit the output on the TL side. Because we cannot easily find a Hebrew→English example for a y-side constituent value constraint, consider the following English→German example (where English is the SL and German is the TL):

```
S::S [NP V NP] -> [NP V NP]
(
...
((Y3 CASE) = ACC)
)
```

In such a case, we would need to learn y-side a constituent value constraint. To see this, note that German NPs mark for case: subjects are generally in nominative case, where objects are generally in accusative case. If we wanted to learn a rule such as the one above, the second German NP (Y3) would have to marked for accusative case, a feature that is not used in English. How can such a constraint be learned? The constraint will be introduced by the Basic Constraints module. The task of the learning module would then be to recognize the cases where a feature is only marked in German, but not in English. In our experiments, this case did not occur when translating from Hebrew into English, but the rule learner provides the functionality for the case that learning is applied to other language pairs. Figure 8.6 summarizes, also in pseudocode, the reconsideration of value constraints.

8.7 Results

8.7.1 Discussion of Learned Rules

In our experiments, we found that a large number of useful constraints were found by the algorithms described above. Some rules are missing some

Advanced Constraints: Retaining value constraints

For all pairs of rules $pair_i = \langle r_1, r_2 \rangle$ s.t. the SL component sequences are the same, and the TL component sequences are different

If $\exists$ a feature f_i s.t. f_i is marked with a value constraint in both rules with different values: $v_{f_i,r_1} \neq v_{f_i,r_2}$

Retain the f_i value constraints in r_1 and r_2

Else remove all value constraints from r_1 and r_2

For all other rules, remove all value constraints

Figure 8.6: Pseudocode for Advanced Constraint Learning: retaining value constraints.

constraints, and some constraints are spurious. Missing constraints make the constrained rules more similar to the context-free rules from the previous chapter. They imply that a larger lattice will be produced, so that the decoder has to choose from a larger number of hypothesis translations. Spurious constraints, on the other hand, are a larger problem, because they will limit the applicability of a rule or prevent correct translations. For this reason, it is preferable to err on the side of missing constraints.

Despite spurious and missing constraints in some rules, we will show in the examples below that the constraints that were learned are useful at run-time.

```
;;SL: H ILD AKL KI HWA HIH R&B
;;TL: THE BOY ATE BECAUSE HE WAS HUNGRY
;;C-Structure:(<S> (<NP> (DET the-1)(N boy-2))
    (<VP> (V ate-3))(<SBAR> (PREP because-4)
    (<S> (<NP> (PRO he-5))(<VP> (V was-6)
    (<ADJP> (ADJ hungry-7))))))
S::S [NP V SBAR] -> [NP V SBAR]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
```

```
(X0 = X2)
((X1 GEN) = (X2 GEN))
((X1 NUM) = (X2 NUM))
((Y1 NUM) = (X1 NUM))
((Y2 TENSE) = (X2 TENSE))
((Y3 NUM) = (X3 NUM))
((Y3 TENSE) = (X3 TENSE))
(Y0 = Y2)
)
```

This rule enforces a number of agreements on the Hebrew side. It enforces agreement between the subject and verb in gender and number. Why is this useful? This will ensure that the rule only applies to cases where the NP and V agree. As Hebrew is highly ambiguous, it can often happen that a sequence of words can be analyzed as a noun and verb, although this is not the correct analysis in this context. Then the above rule would not apply if in the incorrect analysis the noun and verb did not agree in number and gender. The agreement constraints thus ensure that faulty applications are ruled out.

The rule further passes several important features from the SL to the TL. Both sentences must be in the same tense, and the number of the subject is passed. If, for example, the tense feature were not passed from SL to TL, the transfer engine could produce the English verb in all tenses, without preference for any particular one. This would lead to more ambiguity that the decoder must resolve, so that ruling out some incorrect options is useful.

```
;;SL: I$IR B IWTR W P&IL
;;TL: HIGHLY DIRECT AND ACTIVE
;;C-Structure:(<ADJP> (<ADVP> (ADV highly-1))
    (<ADJP> (ADJ direct-2)(CONJ and-3)(ADJ active-4)))
ADJP::ADJP [ADJ ADVP CONJ ADJ] -> [ADVP ADJ CONJ ADJ]
(
(X1::Y2)
(X2::Y1)
(X3::Y3)
(X4::Y4)
(X0 = X2)
((X1 GEN) = (X4 GEN))
((X1 NUM) = (X4 NUM))
(Y0 = Y1)
```

```
)
```

This rule again enforces agreement between SL components. In this case, it is ensured that the first and second adjective agree in number and gender. This is in fact the case in Hebrew. Again, SL constraints help ensure that no incorrect POS sequence analyses result in an incorrect application of the rule.

In this rule, no information is passed from the SL to the TL. This is because adjectives in English do not mark for any features, so that no constraints can or should be learned.

```
;;SL: HIA HGI&H W HWA &ZB
;;TL: SHE ARRIVED AND HE LEFT
;;C-Structure:(<S> (<S> (<NP> (N she-1))
    (<VP> (V arrived-2)))(CONJ and-3)
    (<S> (<NP> (N he-4))(<VP> (V left-5))))
S::S [S CONJ S] -> [S CONJ S]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
(X0 = X3)
((Y1 TENSE) = (X1 TENSE))
((Y3 TENSE) = (X3 TENSE))
(Y0 = Y3)
)
```

This simple rule passes tense information separately for each of the sentences in the conjunction. This can be done because it is known how the components align to each other. The agreement constraints in are exactly in the form that they should be, because it can happen that both sentences are not in the same tense.

```
;;SL: NW$AIM KLLIIM
;;TL: GENERAL SUBJECTS
;;C-Structure:(<NP> (<ADJP> (ADJ general-1))(N subjects-2))
NP::NP [N ADJP] -> [ADJP N]
(
(X1::Y2)
(X2::Y1)
```

```
((X1 GEN) = (X2 GEN))
((X1 NUM) = (X2 NUM))
(X0 = X1)
((Y2 NUM) = (X1 NUM))
(Y0 = Y2)
)
```

Again, this rule encodes a number of useful agreements. On the SL side, it is ensured that the adjective and noun must agree in number and gender (as in previously described rules). Further, only the number feature is transferred to the TL side, because English does not generally mark nouns for gender, as for example in the training example that produced this rule. Nothing is transferred between the aligned adjectives, again because English adjectives do not mark for any features.

```
;;SL: H ILD CRIK LLKT
;;TL: THE BOY MUST GO
;;C-Structure:(<S> (<NP> (DET the-1)(N boy-2))
    (<AUX> (AUX must-3))(<VP> (V go-4)))
S::S [NP AUX V] -> [NP AUX V]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
(X0 = X2)
((X1 GEN) = (X2 GEN))
((X1 NUM) = (X2 NUM))
((Y1 NUM) = (X1 NUM))
(Y0 = Y2)
)
```

As in a previous rule, this rule enforces that the subject and noun in Hebrew agree in number and gender. However, this rule is also an example of a missing constraint. Ideally, this rule would also enforce that the English subject and noun agree in number.

```
;;SL: H ILD AKL TPWX
;;TL: THE BOY ATE AN APPLE
;;C-Structure:(<S> (<NP> (DET the-1)(N boy-2))
    (<VP> (V ate-3)(<NP> (DET an-4)(N apple-5))))
```

Grammar	BLEU	ModBLEU	METEOR
No Grammar	0.0255	0.0910	0.2681
Manual Grammar	0.0713	0.1209	0.3204
Learned Grammar (SeedGen)	0.0281	0.0969	0.2786
Learned Grammar (Compos)	0.0346	0.0998	0.2819
Learned Grammar (MaxCompos)	0.0344	0.0995	0.2811
Learned Grammar (Constraints)	0.0344	0.0993	0.2811

Table 8.2: Automatic evaluation metrics for grammar with constraints.

```
S::S [NP V NP] -> [NP V NP]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
((X2 GEN) = (X3 GEN))
((X2 NUM) = (X3 NUM))
(X0 = X2)
((X1 GEN) = (X2 GEN))
((X1 NUM) = (X2 NUM))
((Y1 NUM) = (X1 NUM))
((Y2 TENSE) = (X2 TENSE))
((Y3 NUM) = (X3 NUM))
(Y0 = Y2)
)
```

This rule is an example of spurious constraints: it enforces that the verb and *object* must agree in number and gender. This is not true in Hebrew. The implication of the spurious constraints will be that this rule will only apply to sentences where the verb and object happen to agree.

8.7.2 Automatic Evaluation Results

As in previous chapters, we here compare the translations on test set 1 for learned grammars with constraints to the translation without a grammar and the translation with a manual grammar. We can observe from Table 8.2 that the expected results were obtained, where the learned grammars with constraints fall in the space between the baseline and the manual grammar on all three automatic evaluation techniques.

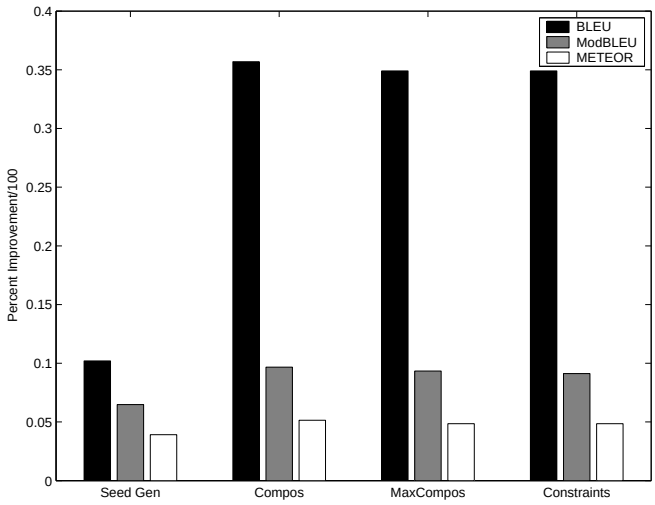

Figure 8.7: Improvement over baseline for test set 1.

Comparison	**BLEU**	**ModBLEU**	**METEOR**
SeedGen	[-0.0073,0.0018]	[-0.0149,0.0026]	p=0.112
Compos	[-0.0321,0.000]	[-0.0164,-0.020]	p=0.084
MaxCompos	[-0.0312,0.000]	[-0.0162,-0.0017]	p=0.108
Constraints	[-0.0304,0.000]	[-0.0162,-0.0018]	p=0.108

Table 8.3: Seed Generation, Basic Compositionality, and Constraint Learning confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline.

Once again we note that the manual grammar outperforms the learned grammar, which can at least in part be attributed to the fact that test set 1 was used as a development set for the manual grammar. The test set used in chapter 9 is unseen both for the manual and the learned grammar, which allows for a fairer comparison. In this case, the performance of the learned grammar matches and sometimes even exceeds the performance of the manual grammar.

As was said above, our goal for Constraint Learning was a reduction in lattice size. For this reason, we report the lattice size for the unconstrained grammar vs. the constrained grammar in Table 8.4. A considerable reduction can be observed, as desired.

Grammar	Lattice Size	Reduction in Size
Learned Grammar, no constraints	87455872	-
Learned Grammar, constraints	72564618	0.170271631

Table 8.4: Lattice sizes for unconstrained and constrained grammars, together with reduction in size over lattice without constraints.

To summarize our findings regarding the performance of constraints, it can be said that grammars with constraints do not generally result in better translation quality. They do however result in a speed-up of the run-time system. In order to achieve higher translation quality *and* a speed-up in run-time, the constraints would have to eliminate only such arcs from the lattice that are 1) poor translations, and 2) would be preferred by the decoder. If an overly general rule results in lattice arcs that are such poor translations that the decoder would not pick them, then Constraint Learning can only hope to achieve an improvement in run-time. This goal was accomplished with the algorithms described here. In section 9 below we will discuss these results in greater detail.

8.8 Case Study: Hebrew Copula

For most of the experiments reported here, we used a structural elicitation corpus for training. This corpus was designed specifically to capture a wide variety of *structural* phenomena. It was, however, not designed for to contain a variety of feature phenomena. When learning constraints, it would be ideal to have a corpus such as

```
I fell.
You fell.
He fell.
She fell.
It fell.
...
```

Learning from such a corpus would allow us, for example, to capture more easily what feature values influence the form of the verb in the minor language SL. More generally, a more feature-oriented corpus would aim at varying feature values in order to determine which values make a difference.

In order to get insight into how the Constraint Learning algorithms proposed in this thesis would perform with a different type of corpus, we chose to perform a case study. The complete feature-based corpus as described in chapter 4 is not yet available. However, we were able to obtain a portion of the corpus that specifically targets copula. A copula is a word that links a subject to its predicate, which can for example be a NP or an ADJP. In English, the verb 'to be' in its various forms acts as a copula in the following sentences:

```
The man was a teacher.
A dog is a kind of dog.
Dogs are nice animals.
He is always impatient.
```

In English, the copula is a verb and takes verb forms. This is not necessarily the case in all languages.

Copulas are interesting to Hebrew→English translation because Hebrew copulas exhibit some properties that are very different from English. We will first briefly describe the issue of copulas in Hebrew, and will then discuss the copula-specific training corpus and the rules that were learned from it.

In Hebrew, present tense copula are often optional. Consider the following example:

```
HWA MWRH
he  teacher.m.s
'He is a teacher.'
```

In this canonical example, no copula is used in the present tense. In the past or future tense, however, the copula is required:

```
HWA HIH       MWRH
he  be.s.past teacher.m.s
'He was a teacher.'

HWA IHIH      MWRH
he  be.m.fut  teacher.m.s
'He will be a teacher.'
```

In the present tense, however, things become more complicated: in some cases, a copula can in fact be used. However, instead of using a form of the verb 'TO BE', Hebrew uses personal pronouns as present tense copulas.

Furthermore, the personal pronouns agree with the subject in number, but not in person. Personal pronouns are used as copulas generally only for definite predicates. A few examples should clarify this phenomenon:

In the following examples, the Hebrew copula can be used, as the predicate 'H TLMID' ('the student') is definite:

```
ATH H TLMID
or
ATH HWA H TLMID
'You are the student.'
```

and

```
ANI H TLMID
or
ANI HWA H TLMID
'I am the student'
```

In the following examples, on the other hand, the copulas are not possible:

```
ANI TLMID
but
*ANI HWA TLMID
'I am a student'
```

```
ATM TLMIDIM
but
*ATM HM TLMIDIM
'You are students'
```

For more details on the complex issue of present tense copula in Hebrew, please refer to (Falk, 2004).

We have created a training corpus of 283 unique elicitation examples for Hebrew copulas. The corpus contains examples such as the ones listed above. We vary **person, gender, number, definiteness (of subject and predicate), tense, and the type of predicate (NP or ADJP)**. A Hebrew informant translated and word-aligned the copula elicitation corpus. In cases where the Hebrew copula is optional, the informant provided both possible translations. The resulting rule learning training corpus contains 319 training examples.

The learned grammar contains 135 rules, and was learned with Maximum Compositionality and all Constraint Learning algorithms. Below, we will discuss some of the resulting rules, and will also discuss what the current Constraint Learning algorithm cannot currently handle, thus pointing to future work.

In order to characterize the learned rules, we will address the following questions:

1. How are past tense copulas handled?
2. How are future tense copulas handled?
3. How are optional present tense copulas handled?
4. Can we distinguish between definite and indefinite predicates for present tense copulas?
5. What features would ideally be there but are not, and why?

Question 1: How are past tense copulas handled?

Below are two typical rules for past tense copula that exemplify well how the grammar handles past tense copula.

In this section, we list the learned rules with all training examples that the rules were learned from. The sentences are listed as SL-TL pairs, where additional training sentences are marked with "alt". For brevity and because they are relatively simple, we do not list the parses for the training examples here.

```
;;SL: H N$IM HIW AIN@LIGN@IWT
;;TL: THE WOMEN WERE INTELLIGENT
;;SL(alt1): H AN$IM HIW AIN@LIGN@IM
;;TL(alt1): THE MEN WERE INTELLIGENT
S::S [NP AUX ADJP] -> [NP AUX ADJP]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
((X2 NUM) = (X3 NUM))
(X0 = X2)
((X1 NUM) = (X2 NUM))
((Y1 GEN) = (X1 GEN))
```

```
((Y1 NUM) = (X1 NUM))
((Y1 PER) = (X1 PER))
(Y0 = Y2)
)
```

In this rule, it is learned that in Hebrew, the subject, copula, and predicate must agree in number, which is correct. It further specifies that gender, number, and person are passed from the Hebrew subject to the English subject. Note that nothing is passed between the adjectives. This is because English adjectives do not mark for anything. The auxiliary transfer will automatically pass lexical-level features from the Hebrew auxiliary to the English auxiliary, so that no constraint is necessary in the rule.

```
;;SL: H N$IM HIW H TLMIDWT
;;TL: THE WOMEN WERE THE STUDENTS
;;SL(alt1): H AN$IM HIW H TLMIDIM
;;TL(alt1): THE MEN WERE THE STUDENTS
S::S [NP AUX NP] -> [NP AUX NP]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
((X2 NUM) = (X3 NUM))
(X0 = X2)
((X1 NUM) = (X2 NUM))
((Y1 GEN) = (X1 GEN))
((Y1 NUM) = (X1 NUM))
((Y1 PER) = (X1 PER))
((Y3 NUM) = (X3 NUM))
((Y3 PER) = (X3 PER))
(Y0 = Y2)
)
```

Some important agreements were learned in this rule. For instance, it is enforced that number, gender, and person values are passed from the Hebrew to the English subject. Notice that gender happens to be marked on the specific English training examples' subjects. Further, it was learned that number and person should be passed from the nominal predicate in Hebrew to the predicate in English. Further, all three Hebrew elements (subject, copula, predicate) must agree in number.

Neither of the two rules specifies that it is particular to past tense. This is because the learning algorithm did not find that it would produce an incorrect context-free sequence in English if it were applied to present tense sentences, and for this reason no tense value constraint was retained. For more details on this issue, please refer to section 8.6.

Question 2: How are future tense copulas handled?

Again, we present two sample rules that were learned for future tense copula.

```
;;SL: H N$IM IHIW AIN@LIGN@IWT
;;TL: THE WOMEN WILL BE INTELLIGENT
;;SL(alt1): H AI$H THIH AIN@LIGN@IT
;;TL(alt1): THE WOMAN WILL BE INTELLIGENT
;;SL(alt2): H AN$IM IHIW AIN@LIGN@IM
;;TL(alt2): THE MEN WILL BE INTELLIGENT
;;SL(alt3): H AI$ IHIH AIN@LIGN@I
;;TL(alt3): THE MAN WILL BE INTELLIGENT
S::S [NP "HIH" ADJP] -> [NP "WILL" "BE" ADJP]
(
(X1::Y1)
;(X2::Y2)
;(X2::Y3)
(X3::Y4)
(X0 = X2)
((Y1 GEN) = (X1 GEN))
((Y1 NUM) = (X1 NUM))
((Y1 PER) = (X1 PER))
(Y0 = Y2)
)

;;SL: H AI$H THIH H TLMIDH
;;TL: THE WOMAN WILL BE THE STUDENT
;;SL(alt1): H AN$IM IHIW H TLMIDIM
;;TL(alt1): THE MEN WILL BE THE STUDENTS
;;SL(alt2): H AI$ IHIH H TLMID
;;TL(alt2): THE MAN WILL BE THE STUDENT
S::S [NP "HIH" NP] -> [NP "WILL" "BE" NP]
(
(X1::Y1)
```

```
;(X2::Y2)
;(X2::Y3)
(X3::Y4)
(X0 = X2)
((Y1 GEN) = (X1 GEN))
((Y1 NUM) = (X1 NUM))
((Y1 PER) = (X1 PER))
((Y4 NUM) = (X3 NUM))
((Y4 PER) = (X3 PER))
(Y0 = Y2)
)
```

Note that in both rules, the copula and the English future tense verb must stay lexicalized, because they are not aligned one-one. This has the side-effect that no constraints are learned for the lexicalized words. This has some implications, because we learn agreement constraints only between heads and their dependents. Since we do not have constraints for the lexicalized words, we also do not have agreement constraints between the copula and the subject or predicate in both cases.

However, we again observe that certain feature values are passed from Hebrew to English. In particular, the English subject receives the gender, number, and person marking from the corresponding Hebrew NP, and the predicate also receives the number and person marking. As in the previous case, no constraints are passed between the adjectives, because English adjectives do not mark for number, person, etc.

Question 3: How are optional present tense copulas handled?

For present tense, there are cases where the copula is used, and cases where it is not used. Consider the following rule:

```
;;SL: ANI TLMIDH
;;TL: I AM A STUDENT
;;SL(alt1): ANI TLMID
;;TL(alt1): I AM A STUDENT
S::S [NP NP] -> [NP "AM" NP]
(
(X1::Y1)
(X2::Y3)
((X1 NUM) = (X2 NUM))
```

```
((Y1 NUM) = (X1 NUM))
((Y1 PER) = (X1 PER))
((Y3 NUM) = (X2 NUM))
((Y3 PER) = (X2 PER))
(Y0 = Y2)
)
```

This rule, among other things, hints at a very important issue that is in fact captured in the rules: for those cases where there is no copula in Hebrew, *only* present tense copulas can be introduced in English. The English copula must remain lexicalized, because it is unaligned. However, the grammar contains rules for no-copula Hebrew sentences only for present tense English copulas. In other words, there are rules similar to the one above that introduce 'IS' and 'ARE', but there are no rules that introduce instead 'WERE'.

This means that the system handles correctly the case of missing Hebrew copula in the present tense.

Another interesting observation can be made about the rule above: although person and number information is passed correctly from the SL to the TL, agreement within the SL is only enforced for person. This is because there is in fact no agreement in person between the two constituents: in both training examples, the subject's person value is 1st, and the predicate's person value is 3rd.

```
;;SL: ANI AIN@LIGN@IT
;;TL: I AM INTELLIGENT
;;SL(alt1): ANI AIN@LIGN@I
;;TL(alt1): I AM INTELLIGENT
S::S [NP ADJP] -> [NP "AM" ADJP]
(
(X1::Y1)
(X2::Y3)
((X1 NUM) = (X2 NUM))
((Y1 NUM) = (X1 NUM))
((Y1 PER) = (X1 PER))
(Y0 = Y2)
)
```

In this rule and the one above, we can again observe that some features are passed from Hebrew to English. Again, the system learns that the Hebrew subject and predicate must agree in number, but not in person.

Although this rule should only apply to first person singular NPs, it is limited in that it only passes number and person information from the SL to the TL, and does not enforce that the person must be 1st and the number singular. This is because the value constraint learner targets constraints only in isolation. *Only* singular or *only* 1st person are not sufficient in deterministically predicting the TL component sequence. In order to learn these two constraints, a different algorithm would be necessary that checks whether *combinations* of value constraints can be useful to predict the TL component sequence. We will return to this issue under question 5 below.

Let us now turn to those cases where Hebrew does indeed use a copula for the present tense. The below rule is one good example.

```
;;SL: H N$IM HN AIN@LIGN@IWT
;;TL: THE WOMEN ARE INTELLIGENT
S::S [NP "HN" ADJP] -> [NP "ARE" ADJP]
(
(X1::Y1)
;--;(X2::Y2)
(X3::Y3)
((X2 GEN) = (X3 GEN))
((X2 NUM) = (X3 NUM))
(X0 = X2)
((X1 GEN) = (X2 GEN))
((X1 NUM) = (X2 NUM))
((Y1 GEN) = (X1 GEN))
((Y1 NUM) = (X1 NUM))
(Y0 = Y2)
)
```

The present tense copulas in Hebrew are not of the same part of speech as the English words they are aligned to: the present tense Hebrew copulas are personal pronouns, whereas the English copulas are as usual auxiliaries. For this reason, the rule learner did not generalize these words to the POS level, and the words remain lexicalized. This has the effect that several rules of the type above are learned, one for each person/gender combination in Hebrew. This simply implies a larger grammar, but has no other serious effects.

As in previous rules, several features are passed from the Hebrew to the English side.

Question 4: Can we distinguish between definite and indefinite predicates for present tense copulas?

For the current system the answer to this question is no.

It is not be possible for the learning algorithms to learn a definiteness feature to distinguish between English outputs, because the learning algorithm described in section 8.6 only allows for the introduction (rather, retention) of a value constraint if a certain feature value would produce a specific English component sequence while a *different* feature value would produce a *different* component sequence consistently. In Hebrew, NPs are indefinite unless marked with the determiner 'H' (or unless their head is a pronoun). Therefore, most Hebrew NPs do not obtain a definiteness feature from any word, so they are simply not marked for definiteness, and no distinguishing feature value can be learned.

Question 5: What features would ideally be there but are not, and why?

We have seen that the learning module was able to capture quite a number, albeit not all, relevant features for Hebrew constraints. This section will address the weaknesses of the current system: why could we not learn certain features, and how can future work address these issues?

The biggest limitation of the current Constraint Learning system is that it deals with features in isolation. For example, it is not able determine that the combination of two or more features leads to a specific translation. For example, consider another present tense copula example:

```
;;SL: H N$IM AIN@LIGN@IWT
;;TL: THE WOMEN ARE INTELLIGENT
;;SL(alt1): H AN$IM AIN@LIGN@IM
;;TL(alt1): THE MEN ARE INTELLIGENT
S::S [NP ADJP] -> [NP "ARE" ADJP]
(
(X1::Y1)
(X2::Y3)
((X1 GEN) = (X2 GEN))
((X1 NUM) = (X2 NUM))
((Y1 GEN) = (X1 GEN))
((Y1 NUM) = (X1 NUM))
((Y1 PER) = (X1 PER))
(Y0 = Y2)
```

```
)
```

Ideally, this rule would include a constraint that would disallow the rule from firing unless the Hebrew subject is marked for plural number or else for 2nd person singular. This could be handled in the following two rules:

```
S::S [NP ADJP] -> [NP "ARE" ADJP]
(
(X1::Y1)
(X2::Y3)
...
((X1 NUM) = P)
...
)
```

and

```
S::S [NP ADJP] -> [NP "ARE" ADJP]
(
(X1::Y1)
(X2::Y3)
...
((X1 NUM) = S)
((X1 PER) = 2)
...
)
```

The current learning system is not able to either split a rule into two or to determine that the combination of two feature values causes a given output. How this can be done automatically is an interesting area for future investigation. In previous work, we have viewed the transfer rules as residing in a version space ((Probst, 2002), (Probst et al., 2003)). In such a framework, we would be able to split the version space into two distinct hypotheses, namely the rules proposed above. Unlike in previous work, this would be possible because we have the actual value constraints with ground values for the components, as we now pass rules from the word level to the constituent level. One possible problem is that in order to do this, we would need to consider all possible combinations of two or more features, greatly increasing the complexity of the learning algorithm.

Another interesting approach to solving this problem has been proposed by Font (Font-Llitjós, 2004). The example rule, together with the training

sentences it was derived from, can be used in the Automatic Rule Refinement module described there. With minimal user feedback, the original rule could be bifurcated and tagged with the appropriate constraints.

Another limitation of the current system is its reliance on the morphology modules. The system has no way of verifying whether the information from the morphology modules is correct and/or complete, so that it must rely on it. Naturally, this can have a negative impact on rule learning. This thesis addresses the task of learning with very limited datasets, so that we cannot afford to eliminate information, even if we do not necessarily consider this information reliable. As we will discuss in section 10.3, one area for future investigation will be to learn from larger datasets. Under such a scenario, we would be able to only accept the information from the morphology modules with a certain probability, and increase the confidence score if the information is found to be consistent with morphological analyses for other words in similar contexts or constructions.

Overall, it was found that the Constraint Learning module is indeed able to handle many phenomena related to copula. The case study also allowed us to pinpoint some weaknesses of the current Constraint Learning algorithms, and we have gained considerable insight into how to address these weaknesses in the future.

Chapter 9

Comprehensive Evaluation

Throughout this document, we have presented aggregate evaluation results of the entire system with our learned grammars on a small test set of 26 sentences that was also used as a development set for the manual grammar. This put the learned grammars at a disadvantage. In this section, we delve deeper into different aspects of the system. The tests in this section will be performed on test sets that are completely unseen both for the manual grammar and for the learned grammars. We will first present a number of ways in which the system can be evaluated. We will then vary a number of parameters and report results, so as to maximize the insight into the performance of the system.

We can view the system as residing in a multi-dimensional space, where the output can depend on a number of settings. We cannot even attempt to explore this space exhaustively. Rather, we will highlight several interesting combinations of settings and report and analyze results for those. We will further put the spotlight on several settings that we think have great potential if explored further, and hope that this will raise concrete research questions for the future.

Thus, the goal of this section is to 1) lay out the space of possible ways to evaluate and further investigate our approach, 2) to report results for some of the area in this space, and 3) to spark interest for promising areas of future research.

9.1 The Evaluation Space

The space that we will lay out here has a number of dimensions, as listed below:

1. **Learning Phases / Settings**: SeedGenOnly, Compositionality, Maximum Compositionality, Constraint Learning, Pruning by rule scoring, etc.

2. **Evaluation**: Automatic evaluation metrics, aggregate vs. rule-based evaluation, effect on run-time

3. **Test Corpora**: Test Set 1, Test Set 2, TestSuite

4. **Run-time Settings**: Lengthlimit

5. **Training Corpora**: Structural corpus vs. comparison corpus, additional data, etc.

It is clearly impossible to evaluate on the cross-product of all these settings. Furthermore, it will not allow us to gain good insight. We will argue why some of these dimensions are of interest to the present thesis, while others are not.

It is clearly interesting to evaluate the different learning settings in order to see what impact they have on the learned rules. It should be kept in mind that the different learning phases often have different goals. In particular, Constraint Learning has two goals: 1) to automatically induce interesting facts about the SL and its relation to the TL, and 2) to constrain the lattice. Constraining the lattice is mostly aimed at reducing the run-time, which can be prohibitively long for longer lengthlimits. The addition of constraints can so lead to longer lengthlimits being feasible at run-time. While adding constraints will generally not have any or much positive effect on the final translation (the decoder is usually strong enough to pick out the best translations in context), the reduction in run-time is in itself a desirable effect if the translation quality does not suffer, or at least does not suffer significantly.

The preceding discussion makes it clear that while we will evaluate different learning phases, they will have to be evaluated from different perspectives, as they have different goals. Thus the second axis, evaluation metrics, will be explored together with different learning phases.

As for the third dimension, test corpora: we will report results on two types of test corpora. The two test corpora that were used up to this point are newspaper text. This kind of evaluation is useful because it shows how our system performs 'in real life', i.e. under harsh conditions of long sentences, uncontrolled vocabulary, etc. The second kind of test corpus that will be used in this chapter is a test suite. The test suite was designed

specifically for this thesis. It comprises a number of sentences that contain structural and linguistic phenomena. Why is this useful? Measuring the performance of our rules on the test suite will allow us to examine what kind of phenomena are captured in our rules.

The fourth dimension deals with different training corpora. One goal of the AVENUE project, within which this research was performed, is the development of training corpora that will be useful for eliciting information about a language in a targeted way. While this is an interesting research question, it is not in the scope of this thesis, so that we will not emphasize the evaluation of our system for different training corpora, and will instead focus on other dimensions. We will however present an ablation study, which will allow us to gain insight of the performance of the system as a function of the training corpus size. We will also present results for a grammar learned from a comparison corpus, which is of the same size as our 'standard' structural corpus, but which was not designed specifically to capture a variety of structural phenomena.

Finally, run-time settings can have a great effect on the performance of the system as a whole. We will briefly present results for different lengthlimits. The lengthlimit is a parameter in the transfer engine that can be set in order to limit the maximum number of input tokens spanned for a given arc. The learned rules are compositional and aimed at combining with each other. The side effect of this combination is that the longer the spanned input becomes, the more combinations of rules are possible, resulting in potentially very large lattices. In fact, the lattices routinely become so large that pruning by setting a lengthlimit is necessary in order for the transfer engine to produce a lattice. The section is aimed at sparking interest for future research: it would be interesting to explore in greater detail the relationship between the lengthlimit and the pruning of the grammar or the introduction of constraints. Again, this is not at the core of this thesis, and is merely presented to promote further research on this issue.

9.2 Overview of Evaluated Settings

In this section, we present an overview of the different system settings that we have evaluated. The following sections will then address each of these settings in turn.

9.2.1 Defaults

The above dimensions will be set to the following defaults:

1. **Learning Phases / Settings**: Seed Generation + Compositionality + Constraint Learning
2. **Evaluation**: Automatic evaluation metrics
3. **Test Corpora**: Test Set 2
4. **Training Corpora**: Structural corpus
5. **Run-time Settings**: Lengthlimit set to 6

The default for the learning phases allows us to evaluate the system as a whole, with all learning phases. Test set 2 is a newspaper text test set of 62 sentences that is unseen for both the manual and the learned grammar, thus allowing for a fairer comparison than test set 1. A lengthlimit of 6 was the highest lengthlimit that the run-time system could handle for the default learned grammar.

9.2.2 Varied Settings

In addition to the default settings, we will vary the first three dimensions, *one at a time*. We will set them to the following:

1. **Learning Phases / Settings**: SeedGenOnly, Compositionality, Rule Scoring / Pruning
2. **Evaluation**: Automatic evaluation metrics, precision, recall of individual rules
3. **Test Corpora**: Test Set 1, Test Set 2, TestSuite

In addition to these variations, we will present a number of settings that 'fall off the grid'. One of these settings is an evaluation of the system using different training corpora. We present a section on training a grammar with different training corpora. This will allow us to gain insight into what can be achieved when the corpus is not as small as our standard training corpus. In a bigger corpus, the problem of "missing" structures diminishes. At the same time, the grammars become very big, leading to large lattices. This evaluations is largely intended as motivation for future work. In this thesis, we concentrated on small corpora, and how much we could learn from them. Future work could address the learning of rules from larger corpora, as with larger corpora some approaches need to be revised. For instance,

pruning will become much more important, as larger corpora result in larger grammars and thus in higher ambiguity and bigger lattices.

Another evaluation is a series of different lengthlimits. It is expected that a longer lengthlimit will result in higher scores, as shorter lengthlimits cause rules or rule combinations not to fire.

We finally explore the possibility of scoring rules in order to prune the grammar. The transfer engine does not explicitly handle scored grammar rules; it will produce a full lattice given a grammar and a lexicon. To reduce the lattice, we are forced to reduce the grammar. An important issue to note here is that there is a trade-off between what the decoder can pick out of a large lattice, and producing a smaller lattice. The trade-off is very similar to the trade-off with lengthlimits above (and actually very similar to the trade-off with introducing constraints). However, when pruning the grammar, we also hope to extract those rules that are powerful and accomplish good translation power, and to discard 'bad'. rules. If bad rules are eliminated, we not only achieve a run-time speed-up (because the resulting lattice will be smaller), but we also eliminate those arcs from the lattice that are undesirable. Given the power of the decoder to extract the best arcs from the lattice, it is hard to prune the grammars reliably and to accomplish an increase in performance. For this reason, the main goal is to reduce the run-time without a big reduction in performance.

Once again we note that often the results that show a big improvement in translation quality are not highly statistically significant, because the test set size is small (62 sentences). We therefore report significance tests mostly for completeness, and do not attach too much weight to them.

9.3 Default Setting Evaluation

In the default evaluation, we used the Compositionality setting with the assumption of Maximum Compositionality. The lengthlimit was set to 6 for all tests. The learned grammar was tested on the test set 2 (62 sentences), and the system's performance was compared to the performance of the system using the manual grammar and no grammar. The system without a grammar consists of lexical transfer only, i.e. translation using the lexicon, but no grammar rules. The partial translations produced by the lexicon are then entered into a lattice and decoded exactly as if they had been produced by a lexicon *and* a grammar. The manual grammar was used with the default lengthlimit of 6. We report BLEU, ModBLEU, and METEOR scores in Table 9.1. We furthermore illustrate the improvement over baseline in

graphical form in Figure 9.1.

Grammar	BLEU	ModBLEU	METEOR
No Grammar	0.0565	0.1362	0.3019
Manual Grammar	0.0817	0.1546	0.3241
Learned Grammar	0.0780	0.1524	0.3293

Table 9.1: Evaluation results for default settings on test set 2 for Hebrew→English translation.

Comparison	BLEU	ModBLEU	METEOR
Learned default	[-0.0386,0.0048]	[-0.0263,-0.0070]	p=0.050

Table 9.2: Default settings confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline.

It can be seen from the results that our approach results in significant gains in translation performance. The percent improvement depends in part on what evaluation metric is used. However, for all three evaluation metrics the gain is considerable. It can also be observed that the METEOR score assesses the translation quality of the fully automatic system as higher than with the manually written grammar. This is a very positive result for our approach.[1]

9.4 Varying Learning Settings

In this section, we present a comparative evaluation of the system under different conditions: we learn several grammars with different parameter settings, using different parts of the learning system. We report the following scores: BLEU, Modified BLEU, and METEOR. In Table 9.3 we report the results for different learning settings for the test set 2.

In the first evaluation, we compare a learned grammar that only uses the Seed Generation module to a system with no grammar and to a system using the manual grammar. Seed Generation only results in flat, non-compositional rules. This means that each of the rules can only apply in

[1]In earlier chapters, we reported evaluation results on test set 1, where the manual grammar performed better than the learned grammar. We believe that this is in part due to the fact that the manual grammar was designed using this first set, test set 1.

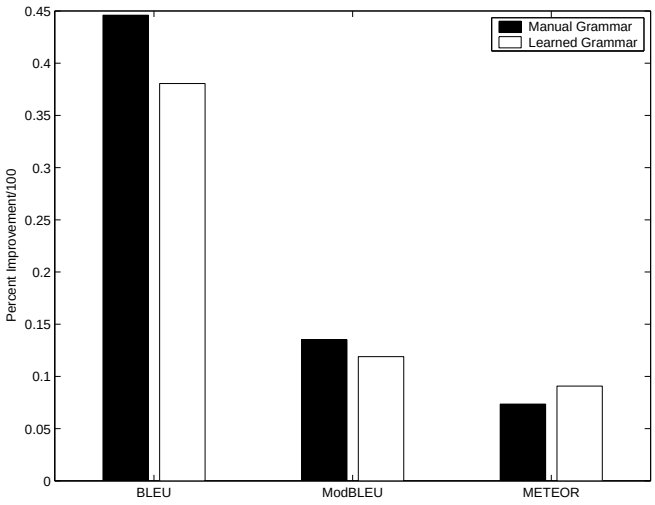

Figure 9.1: Improvement over Baseline for Default Settings on Hebrew→English translation.

Grammar	**BLEU**	**ModBLEU**	**METEOR**
No Grammar	0.0565	0.1362	0.3019
Manual Grammar	0.0817	0.1546	0.3241
Learned Grammar (SeedGen)	0.0741	0.1498	0.3239
Learned Grammar (MaxCompos)	0.0772	0.1519	0.3297
Learned Grammar (Constraints)	0.078	0.1524	0.3293

Table 9.3: Evaluation results for different learning settings on test set 2.

isolation. The rules are often more lexically bound and thus more specific (i.e. apply to fewer contexts). However, they generalize from lexical items to the POS level wherever possible, which allows them to capture important transfers. Table 9.3 shows the results for this experiment for the test set 2.

It can be seen that Seed Generation alone results in a system consistently above the baseline system. This indicates that much can be learned from copying the flat structure of the training sentences into the rules. It can be concluded that the training data exhibits a number of frequent and important transfers. If we can get significant gain out of this process, we can conclude that the POS sequences play a very important role in determining the syntactic structure of a sentence, because the rule applications result in

Comparison	BLEU	ModBLEU	METEOR
SeedGen	[-0.0343,-0.0020]	[-0.0229,-0.0049]	p=0.148
MaxCompos	[-0.0379,-0.0044]	[-0.0256,-0.0063]	p=0.049

Table 9.4: Different learning settings: confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline.

correct word reorderings.

In the next experiment, we ran the rule learning using the Seed Generation *and* Compositionality, meaning that we learn a set of compositional rules that do not have any constraints. The advanced techniques described in the section 7.6 were used for this experiment, but they are rare and do not impact the score. Table 9.3 shows the results for this experiment for the test set 2, again comparing the learned grammars to the baseline (no grammar) and the manually written grammar.

Once again, the system performs well above baseline, an encouraging result. When comparing to Seed Generation only, we see that the more general grammar in fact performs better than Seed Generation only.

Finally, in Table 9.3 we also report evaluation results for a grammar with constraints. In the above results it can be observed that the constraints did not result in improved translation performance. As has been discussed in previous chapters, the goal of Constraint Learning is a reduction in run-time. This is accomplished by preventing the transfer engine from producing a large number of incorrect ambiguities, which result in additional arcs in the lattice. When constraints are added to the grammar, some ambiguities can successfully be eliminated. As was discussed before, in order to achieve higher translation quality *and* a speed-up in run-time, the constraints would have to eliminate only such arcs from the lattice that are 1) poor partial translations, and 2) would be preferred by the decoder. If an overly general rule results in lattice arcs that are such poor translations that the decoder would not pick them, then Constraint Learning can only hope to achieve an improvement in run-time. This goal was accomplished with the algorithms described here.

As was mentioned above, BLEU and Modified BLEU emphasize measuring precision, whereas the METEOR score emphasizes recall. From this perspective, the results on the learned grammar with constraints can be explained: adding constraints to the grammar results in smaller lattices, as can be seen in Table 9.5. Furthermore, the processing time is reduced, as is

shown in tables 9.6 and 9.7.

Grammar	**Lattice Size**	**Reduction in Size**
Learned Grammar (MaxCompos)	187275514	-
Learned Grammar (Constraints)	149713589	0.20057

Table 9.5: Lattice sizes for unconstrained and constrained grammars, together with reduction in size over lattice without constraints.

Grammar	**System Time**
Learned Grammar (MaxCompos)	54.98
Learned Grammar (Constraints)	33.28

Table 9.6: Lattice creating times on test set 2 (system time in seconds) for grammars with and without constraints.

Grammar	**System Time**
Learned Grammar (MaxCompos)	3123.38
Learned Grammar (Constraints)	2287.47

Table 9.7: Decoding times on test set 2 (system time in seconds) for grammars with and without constraints.

The speed-up in processing time comes at the cost of some recall, while compromising hardly any precision. It would be hoped that adding constraints would increase in more improved precision (i.e. higher BLEU and Modified BLEU) scores. However, the speed-up in processing power, especially during decoding, still very much justify introducing constraints. Lastly, constrained rules can feed into a rule refinement module as proposed by Font-Llitjós (Font-Llitjós, 2004), which in part aims at improving the accuracy of unification constraints, in particular for automatically learned rules.

9.5 Varying Evaluation - Rule Level Evaluation

The goal of this section is to evaluate the performance of individual rules. This has two goals: 1) we can inspect the highest-performing rules, thus gain-

ing insight into what most causes the increase in translation performance, and 2) rule scoring allows us to prune from the grammar the lowest-scoring rules.

Rule scoring is done via a lattice scoring method that is described in the appendix section A. This method assigns individual scores to each rule in the grammar. With precision scores at hand, we can prune the grammar by eliminating the low-scoring rules. This allows us to eliminate the worst-scoring rules, with the goal of producing a higher-quality grammar. Rule-level scoring and subsequent pruning can be seen as a post-learned filter for the learned rules. We first learn a set of rules, and then eliminate those that do not perform well in practice. For example, if a rule was learned from noisy data, it would perform poorly at run-time, and should be eliminated. As was said above, it is hard for this technique to yield an increase in translation performance overall, because the filter would need to automatically eliminate 1) rules that produce bad arcs that 2) the decoder would choose for the final translation. For this reason, a speed-up at run-time is a more achievable goal. Run-time is generally improved by this technique, because a pruned grammar almost necessarily leads to smaller lattices, which was in fact observed in our experiments. If pruning results additionally in improved translation power, then two goals are accomplished at the same time.

In the experiments reported in this section, we first obtained precision scores based on test set 1, which was used for evaluation in previous chapters. We then resorted the grammar based on the individual rule scores. This allowed us to eliminate a portion of the grammar, leaving only 25%, 50%, 75%, or the complete grammar. Then we can assess the quality of the pruned grammar on test set 2, which is unseen data to the grammars.

The rules were scored using a lattice evaluation metric to assign individual scores to arcs. The lattice scoring metric is described in detail in the appendix in section A. Without getting into the details here, the metric assigns scores to individual arcs in a manner that is very similar to the BLEU and METEOR scores: it compares unigrams, bigrams, trigrams, ... n-grams to the reference translation. The more unigrams, bigrams, etc. match the reference translation, the higher the score of the arc.

Because we know what rule or rules were applied to produce a specific arc, we can then compute a precision score for each rule. Each arc is assigned to the rule which was the top-level rule in the arc production. For example, if an arc was produced by a sentence-level rule $S, 1$, which filled its subject with an NP rule $NP, 2$, then this specific arc is only assigned only to $S, 1$, because it is the top-level rule for this arc production. The scor-

ing mechanism abstracts away from mistakes that are made by lower-level rules also involved in the arc production: for a given span of indices and a given rule, only the highest-scoring arc is assigned to the top-level rule. This eliminates problems caused by lower-level rules as well as ambiguous lexical selection.

For each individual rule, we sum the arc scores for all arcs 1) to which the rule contributed as the top-level rule, and 2) which have the maximum score for their span. We then divide by the number of rule applications in order to get a confidence (precision) score for the rule. This precision score captures the average quality of an arc to which the given rule contributed as the top-level rule. If we denote with k the number of arcs to which rule R_i contributed, and we denote the subset of arcs in the lattice to which rule R_i contributed as $ArcSet_{R_i}$ (i.e. $|ArcSet_{R_i}| = k$), then the rule precision is:

$$RulePrecision = \frac{\sum_{j=1}^{k} ArcScore_j}{k}, j \in ArcSet_{R_i}$$

Since the arc scores are normalized (between 0 and 1), the precision score can easily be interpreted as the power of the rule to produce translations that are close to the reference translation.

It should be emphasized here that the lattice scoring metric as described in section A is only possible because the transfer engine and the decoder are two separate software systems. The transfer engine produces a full lattice, which then serves as input to the decoder. The rule level evaluation analyzes the individual arcs (i.e. partial translations) in the lattice. In the appendix section A, we describe a possible approach to rule scoring under a scenario where the transfer engine and decoder are not separate modules.

If we were not able to analyze the individual arcs, for instance if the transfer engine and the decoder were merged into an integrated step, then the rule level evaluation method would need to be modified. In such a case, we could assess the quality of individual rules as follows: in order to asses the quality of rule R_i in grammar G, use grammar G to translate a development set, or as here the training set. This results in a translation score for G, $score_G$. Then eliminate R_i from G, resulting in modified grammar $G' = G \setminus \{R_i\}$. Run G' on the same development or training set to obtain $score_{G'}$. The quality of R_i can then be estimated by $score_G - score_{G'}$. This method would be an effective way of estimating rule quality. Its drawback is efficiency: especially for large grammars, the operation would be expensive, because the training or development set must be translated separately to assess the quality of each rule. For this reason, the lattice scoring metric

used here is preferable in practice.

The following table, Table 9.8, lists the rules with the highest precision score according to the lattice scoring.

PP,7	0.055634892
NP,25	0.04240812
NP,10	0.03525641
NP,21	0.034090909
NP,17	0.033509862

Table 9.8: Rules with highest rule precisions.

```
{PP,7}
;;SL: &BWR AW@WBWSIM W &BWR MKWNIWT
;;TL: FOR BUSES AND FOR CARS
;;C-Structure:(<PP> (<PP> (PREP for-1)(<NP> (N buses-2)))
                    (CONJ and-3)
                    (<PP> (PREP for-4)(<NP> (<NP> (N cars-5)))))
PP::PP [PP CONJ PP] -> [PP CONJ PP]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
(X0 = X1)
((Y1 NUM) = (X1 NUM))
((Y3 NUM) = (X3 NUM))
(Y0 = Y1)
)

{NP,25}
;;SL: KBI$IM MHIRIM W &BWDWT CIBWRIWT AXRWT
;;TL: HIGHWAYS AND OTHER PUBLIC PROJECTS
;;C-Structure:(<NP> (<NP> (N highways-1))
                    (CONJ and-2)
                    (<NP> (<ADJP> (ADJ other-3))
                        (<ADJP> (ADJ public-4))(N projects-5)))
NP::NP [NP CONJ NP] -> [NP CONJ NP]
(
(X1::Y1)
```

```
(X2::Y2)
(X3::Y3)
(X0 = X3)
((Y3 NUM) = (X3 NUM))
(Y0 = Y3)
)

{NP,10}
;;SL: @MPR@WRH W LXC
;;TL: TEMPERATURE AND PRESSURE
;;C-Structure:(<NP> (N temperature-1)(CONJ and-2)(N pressure-3))
NP::NP [N CONJ N] -> [N CONJ N]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
(X0 = X3)
((Y1 NUM) = (X1 NUM))
((Y3 NUM) = (X3 NUM))
(Y0 = Y3)
((Y1 PERS) = (Y3 PERS))
)

{NP,21}
;;SL: BNINIM CIBWRIIM XD$IM
;;TL: NEW PUBLIC BUILDINGS
;;C-Structure:(<NP> (<ADJP> (ADJ new-1))(<ADJP> (ADJ public-2))
                                                (N buildings-3))
NP::NP [N ADJP ADJP] -> [ADJP ADJP N]
(
(X1::Y3)
(X2::Y2)
(X3::Y1)
((X1 GEN) = (X3 GEN))
((X1 NUM) = (X3 NUM))
((X2 GEN) = (X3 GEN))
((X2 NUM) = (X3 NUM))
(X0 = X1)
((Y3 NUM) = (X1 NUM))
(Y0 = Y3)
```

```
)

{NP,17}
;;SL: TKNIT H @IPWL H HTNDBWTIT
;;TL: THE VOLUNTARY CARE PLAN
;;C-Structure:(<NP> (DET the-1)(<ADJP> (ADJ voluntary-2))
                                    (N care-3)(N plan-4))
NP::NP [N "H" N "H" ADJP] -> ["THE" ADJP N N]
(
(X1::Y4)
;(X2::Y1)
(X3::Y3)
;(X4::Y1)
(X5::Y2)
(X0 = X1)
((Y3 NUM) = (X3 NUM))
((Y4 NUM) = (X1 NUM))
(Y0 = Y4)
)
```

The first three rules are straightforward in the sense that they do not reorder any of the constituents. However, their importance lies in the fact that they combine words in conjunctions, which can then, as a unit, be used in higher-level rules. The fourth rule is an interesting in that it captures the reordering of adjectives and nouns when translating from Hebrew to English. The fifth-ranking rule is similar, but it captures the facts that 1) Hebrew adjectives occur after the noun, but before the noun in English, 2) in Hebrew noun compounds, definiteness is marked on the second noun, while in English it is marked before both nouns, and 3) Hebrew adjectives are marked for definiteness, while English adjectives are not.

Some rules do not apply at run-time. This can have several reasons. The most prominent is that we are forced to apply a lengthlimit during lattice creation. With the lengthlimit set to 6, some rules cannot apply because they span more than 6 source language indices. This complication is hard to overcome. One possible area for future work is to make the lengthlimit more flexible. More specifically, the lengthlimit generally prevents repeated applications of the same rule to the same arc, resulting in combinatorial explosion. In the future, we can develop a method whereby the lengthlimit is enforced only if the same rule applies repeatedly, or if the applying rule does not have a high precision score.

As was said in the leading paragraph of this section, part of the rationale behind rule scoring is the possibility to prune the grammar. In the below experiments, we pruned the grammar to 25%, 50%, and 75% of the original size, and report results on test set 2 (table 9.9). It can be seen that pruning the grammar to 75% actually yields slightly improved translation performance, which was a hard goal to accomplish. The results for the grammar pruned to 75% are highly statistically significant on this test when comparing the learned grammar to the baseline.

Grammar	**BLEU**	**ModBLEU**	**METEOR**
No Grammar	0.0565	0.1362	0.3019
Manual Grammar	0.0817	0.1546	0.3241
Learned Pruned (25%)	0.0565	0.1362	0.3019
Learned Pruned (50%)	0.0592	0.1389	0.3075
Learned Pruned (75%)	0.0800	0.1533	0.3296
Learned Pruned (full)	0.078	0.1524	0.3293

Table 9.9: Evaluation results for grammars pruned to different sizes.

Comparison	**BLEU**	**ModBLEU**	**METEOR**
Pruned (25%)	[N/A]	[N/A]	[N/A]
Pruned (50%)	[-0.0104,0.0034]	[-0.0063,0.0004]	p=0.121
Pruned (75%)	[-0.0402,-0.0092]	[-0.0270,-0.0089]	p=0.023

Table 9.10: Pruned grammars: confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline. No p-value and confidence intervals could be obtained for the grammar pruned to 25%, because the values are the same for the learned grammar and the baseline.

We also report the run-time lattice creation (Table 9.11) and decoding (Table 9.12). We expected that run time would be reduced drastically at some cost in score, which was supported by the results except for the grammar pruned to 75%, which resulted in a slight performance increase. In Table 9.11, we further report the reduction in lattice size over the lattice for the full grammar. We observe a sizeable reduction already for the grammar that was reduced to 75% of the full grammar.

Grammar	**System Time**
Learned Grammar (25%)	1.02
Learned Grammar (50%)	1.81
Learned Grammar (75%)	5.91
Learned Grammar (full)	33.28

Table 9.11: Lattice creating times on test set 2 (system time in seconds) for grammars pruned to different sizes.

Grammar	**System Time**
Learned Grammar (25%)	22.55
Learned Grammar (50%)	189.89
Learned Grammar (75%)	397.06
Learned Grammar (full)	2287.47

Table 9.12: Decoding times on test set 2 (system time in seconds) for grammars pruned to different sizes.

9.6 Varying Test Corpora - Test Suite

The test suite is a test set of 119 sentences that was designed to capture different linguistic structures, so that it can be assessed whether the learned grammar can handle these structures. The test suite was designed in English and translated by one informant into Hebrew. The following structures are tested in the test suite:

- Subordinate clauses
- Conjunction of PPs
- Adverbs
- Direct objects
- Reordering of adjectives and nouns
- Pro-drop
- Comparatives and superlatives
- NP→ NP PP

Grammar	Lattice Size	Reduction in Size
Learned Grammar (25%)	330342	0.997793507
Learned Grammar (50%)	13431206	0.910287329
Learned Grammar (75%)	29242597	0.804676401
Learned Grammar (100%)	149713589	-

Table 9.13: Lattice sizes for full and pruned grammars, together with reduction in size over lattice for full grammar.

- ADJ ADJ
- ADJP embedded in NP

The following table (Table 9.14) shows the results; Table 9.15 lists the confidence intervals for BLEU and ModBLEU and the p-value for METEOR.

Grammar	BLEU	ModBLEU	METEOR
No Grammar	0.0746	0.1562	0.4146
Manual Grammar	0.1179	0.1887	0.4471
Learned Grammar	0.1199	0.1914	0.4655

Table 9.14: Evaluation results for different learning settings on test suite.

Comparison	BLEU	ModBLEU	METEOR
Comparison Corpus	[-0.0706,-0.0243]	[-0.0495,-0.0215]	p=4.32327E-05

Table 9.15: Test suite confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline.

It can be observed that the learned grammar performs very well, well above baseline. In addition, the learned performs even above the manual grammar on all scores, and with a considerable margin for the METEOR score. Both of these scores emphasize recall more than BLEU and ModBLEU do, once again hinting at the higher generality of the learned grammar.

In Table 9.2, we show the percent improvement over the baseline for the test suite. A complete listing compares of the differences in translation when the learned grammar was used versus when no grammar was used can

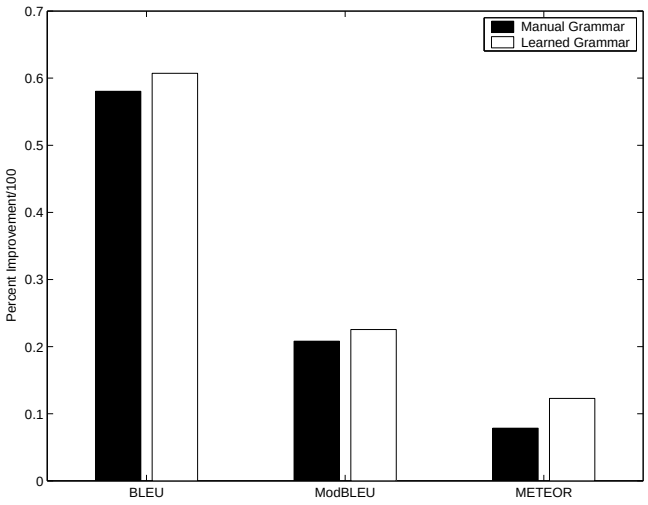

Figure 9.2: Improvement over baseline for test suite.

be found in the appendix in section B. Below, we list examples of each of the phenomena that were targeted in this test suite. Clearly, the test suite contains many more phenomena than the ones listed above (e.g., simple NPs, etc.). We chose to concentrate our examination on those phenomena listed above, because they are interesting structures that we can hope to capture with grammar rules, and where a simple word-by-word translation would fail to produce the correct structure.

Subordinate clauses:

No Grammar:	i slept when he read you the tablet
Learned grammar:	i slept when he read the book
Reference translation:	I slept while he read the book

Conjunctions of PPs:

No Grammar:	students youths on the road all want to learn countries learn and for people learn
Learned Grammar:	young students with access all wish to study various countries and with different peoples
Reference Translation:	Young students often want to learn about different countries and about different people.

Adverbs:

No Grammar:	the drug improved slowly you health his
Learned Grammar:	the drug improved slowly you his health
Reference Translation:	The drug slowly improved his health

Direct objects:

No Grammar:	the mother miss for child of her
Learned Grammar:	the mother miss for her son
Reference Translation:	The mother misses her child

No Grammar:	the president promised support faster to assemble
Learned Grammar:	the president promised stronger support to assemble
Reference Translation:	The President promised stronger support for the army

Reordering of adjectives and nouns:

No Grammar:	the students the successful transfer you the examination
Learned Grammar:	the successful students study you the examination
Reference Translation:	The good students passed the exam

No Grammar:	the house the beautiful near river sell for 100000 dollars
Learned Grammar:	the beautiful home near river sell for 100000 dollars
Reference Translation:	The pretty house near the river was sold for $ 1000000

Pro-drop:

No grammar:	we went to museum
Learned grammar:	we went to museum
Reference translation:	We went to the museum
Comment:	system without grammar also produces correct translation because of enhanced lexicon

Comparatives:

No Grammar:	he suggested to him contract longer
Learned Grammar:	he suggested if longer treaty
Reference Translation:	He offered him a longer contract

NP→ NP PP:

No Grammar:	the decision the final by lot his for the captain cpwih become known in $bwa show
Learned Grammar:	the final decision on his fate of the captain cpwih become known in $bwa show
Reference Translation:	The final decision on the captain 's fate is expected will be announced publicly next week

ADJ ADJ (plus reordering in NP):

No Grammar:	we coat for communication public grand today
Learned Grammar:	we coat for important public announcement today
Reference Translation:	We expect an important public announcement today

ADJPs embedded in NPs:

No Grammar:	the film show you the fact the tough greatly for the persons in the country the it
Learned Grammar:	the film show you the very harsh reality of the adults in the country the it
Reference Translation:	The movie shows the extremely difficult situation of the people in this country

9.7 Varying Run-Time Settings - Lengthlimits

As discussed above, the transfer engine can be run in a setting that restricts the maximum arc length that is produced. This means that when the lengthlimit is set to n, only arcs spanning up to n input segments are produced. 'Segments' are generally considered to be words. For the Hebrew system, the morphology step divides each word into one or more segments. For example, 'HBIT' ('THE HOUSE') is split by the morphology module into two segments, one for the definite determiner ('H'), and one for the noun ('BIT'). In this sense, lengthlimit is defined slightly differently within the transfer engine for Hebrew than for other languages. However, the goal of this section is to examine the relative impact of different lengthlimits on the score. In general, it is true that the longer the lengthlimit, the more the rules can apply and combine. However, a longer lengthlimit also implies a large number of ambiguities.

The following tables and figures compare the system with no grammar, with an automatically learned grammar, and with the manual grammar under different lengthlimits. As elsewhere, we report BLEU (Tables 9.17

Lengthlimit	**1**	**2**	**3**	**4**	**5**	**6**
No Grammar	0.019	0.0521	0.0535	0.056	0.0565	0.0565
Manual Grammar	0.0221	0.056	0.0697	0.0784	0.0806	0.0817
Learned Grammar	0.019	0.0562	0.0588	0.0763	0.0772	0.078

Lengthlimit	**7**	**8**	**9**
No Grammar (cont.)	0.0565	0.0565	0.0565
Manual Grammar (cont.)	0.0822	0.0854	0.0854
Learned Grammar (cont.)	-	-	-

Table 9.17: BLEU scores for different lengthlimits.

Lengthlimit	**1**	**2**	**3**
Learned Grammar	[N/A]	[-0.0806,0.0437]	[-0.0166,0.0039]

Lengthlimit	**4**	**5**	**6**
Learned Grammar (cont.)	[-0.0380,-0.0031]	[-0.0388,-0.0044]	[-0.0386,0.0048]

Table 9.18: Different lengthlimits: confidence intervals for BLEU. No confidence interval could be obtained for a lengthlimit of 1, because the values are the same for the learned grammar and the baseline.

and 9.18 and Figure 9.3), ModBLEU (Tables 9.19 and 9.20 and Figure 9.4), and METEOR (Tables 9.21 and 9.22 and Figure 9.5) scores. For the system without a grammar and with the manual grammar, we were able to report results for lengthlimits up to 9. For the learned grammar, as was said above, the maximally possible lengthlimit was 6.

It is interesting to note that the system without a grammar (lexicon only) experiences gains in performance up to a lengthlimit of 3. This is explained by two phenomena: many entries in the lexicon have a SL side consisting of more than one token. These entries are not usable when lengthlimit is set to 1. The second reason returns to the Hebrew segmentation issue addressed above: a given Hebrew input word (as delimited by spaces) can be split into multiple segments by the morphology module. If so, this word could not be translated in parts by the system if the lengthlimit is set to 1. It could only be translated reliably if the correct morphological analysis does not result in multiple segments. This effect tapers off at a lengthlimit of 3. This is not unexpected, as most words are analyzed into one or two segments, and most dictionary entries have either one or two tokens on the SL side.

As for the systems that use the learned and manual grammar, respec-

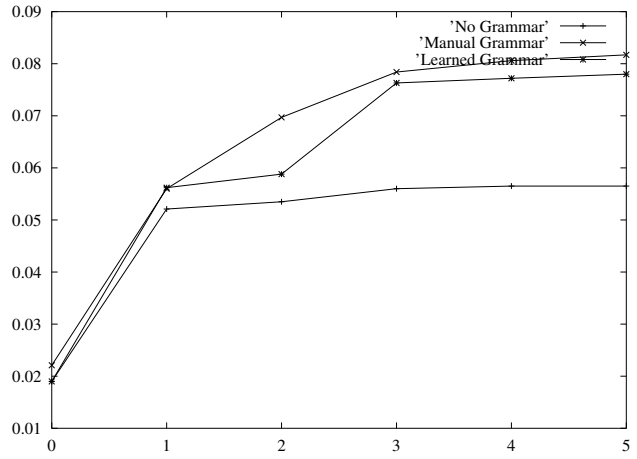

Figure 9.3: BLEU scores across different lengthlimits, comparing no grammar, manual grammar, and learned grammar.

Lengthlimit	**1**	**2**	**3**	**4**	**5**	**6**
No Grammar	0.0676	0.1287	0.1322	0.1354	0.1362	0.1362
Manual Grammar	0.069	0.1297	0.1429	0.1517	0.1539	0.1546
Learned Grammar	0.0676	0.1311	0.1352	0.1498	0.1518	0.1524

Lengthlimit	**7**	**8**	**9**
No Grammar (cont.)	0.1362	0.1362	0.1362
Manual Grammar (cont.)	0.1552	0.1568	0.1568
Learned Grammar (cont.)	-	-	-

Table 9.19: Modified BLEU scores for different lengthlimits.

Lengthlimit	**1**	**2**	**3**
Learned Grammar	[N/A]	[-0.0074,0.0019]	[-0.0076,-0.0012]

Lengthlimit	**4**	**5**	**6**
Learned Grammar (cont.)	[-0.0239,-0.0052]	[-0.0255,-0.0067]	[-0.0263,-0.0070]

Table 9.20: Different lengthlimits: confidence intervals for ModBLEU. No confidence interval could be obtained for a lengthlimit of 1, because the values are the same for the learned grammar and the baseline.

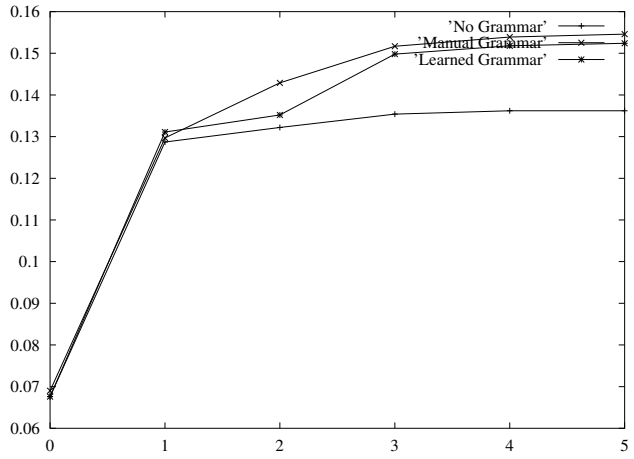

Figure 9.4: Modified BLEU scores across different lengthlimits, comparing no grammar, manual grammar, and learned grammar.

Lengthlimit	**1**	**2**	**3**	**4**	**5**	**6**
No Grammar	0.171	0.2962	0.3016	0.3012	0.3019	0.3019
Manual Grammar	0.1744	0.297	0.3141	0.3182	0.3232	0.3241
Learned Grammar	0.171	0.2995	0.3072	0.3252	0.3282	0.3293

Lengthlimit	**7**	**8**	**9**
No Grammar (cont.)	0.3019	0.3019	0.3019
Manual Grammar (cont.)	0.3257	0.3276	0.3276
Learned Grammar (cont.)	-	-	-

Table 9.21: METEOR scores for different lengthlimits.

Lengthlimit	**1**	**2**	**3**
Learned Grammar	[N/A]	p=0.302	p=0.377

Lengthlimit	**4**	**5**	**6**
Learned Grammar (cont.)	p=0.086	p=0.067	p=0.050

Table 9.22: Different lengthlimits: confidence intervals for METEOR. No p-value could be obtained for a lengthlimit of 1, because the values are the same for the learned grammar and the baseline.

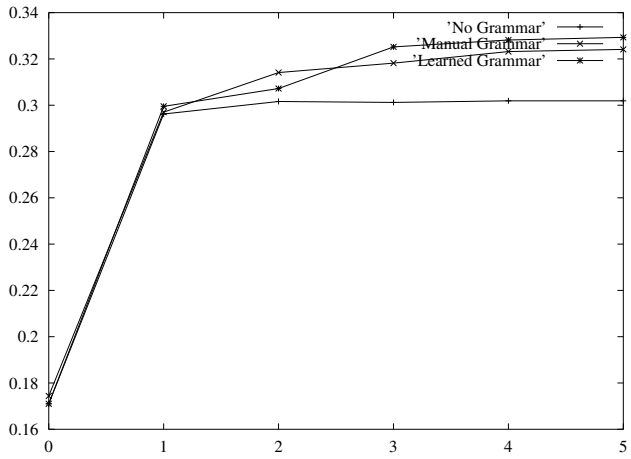

Figure 9.5: METEOR scores across different lengthlimits, comparing no grammar, manual grammar, and learned grammar.

tively, it is observed that the performance increases as a function of the increasing lengthlimit. This is not surprising because for longer lengthlimits more rules can apply and combine. However, it should be expected that the performance tapers off with increasing lengthlimits as the increasing number of ambiguities outweighs the benefit of adding new good arcs to the lattice when more rules can apply, and as the lengthlimit exceeds the length of the learned rules. Most transfer rules are learned from relatively short examples, so that there are no transfer rules for extremely long phrases (such phrases can however still be covered by combinations of rules). Although the argument of short training examples does not hold for the manual grammar, the effect of translation quality tapering off is also observed with the manual grammar: moving from a lengthlimit of 5 to a lengthlimit of 6 results only in negligible performance gain. The performance boost continues to be very gradual for lengthlimits higher than 6.

9.8 Varying Training Corpora

9.8.1 Comparison Corpus

So far, we have focused on the results using a training corpus of 120 training elements. This corpus was specifically designed to be structurally diverse,

which is expected to facilitate learning. In order to test this hypothesis, we also ran an experiment where we learned a grammar from a corpus of equivalent size (120 training examples), but with less structural diversity. We took as the training corpus the first part of the original functional corpus.[2] This corpus contains a number of sentences and phrases of low complexity, as for example:

```
I lost a tooth.
A star was shining brightly.
a car
with his finger
```

These training examples lend themselves to rule learning in a way similar to the examples in the structural corpus. However, the structural diversity in this corpus is more limited than in the structural corpus. Therefore, taking 120 phrases and sentences from this corpus provides a good basis for comparison to test our hypothesis: does it help rule learning to create a corpus that is structurally diverse? Table 9.23 summarizes our findings (p-value and confidence intervals in Table 9.24).

Grammar	**BLEU**	**ModBLEU**	**METEOR**
No Grammar	0.0565	0.1362	0.3019
Manual Grammar	0.0817	0.1546	0.3241
Learned Grammar (Structural)	0.0780	0.1524	0.3293
Learned Grammar (Comparison)	0.0626	0.1412	0.3123

Table 9.23: Results comparing the Structural Corpus to a comparison corpus of equal size.

Comparison	**BLEU**	**ModBLEU**	**METEOR**
Comparison Corpus	[-0.0160,0.0021]	[-0.0097,-0.0005]	p=0.345

Table 9.24: Comparison corpus confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline.

[2]Note that this is not the copula corpus, but rather the functional corpus described in (Probst & Levin, 2002).

The results indeed support our hypothesis. The structural corpus leads to a stronger grammar that can perform better on unseen data. However, it was also found that the comparison corpus, in which structural diversity was not a high priority, outperforms the baseline.

9.8.2 Additional Training Data

It has been emphasized repeatedly that the goal of the algorithms developed here is to optimize learning from a very small training corpus. In particular, the Compositionality algorithms aimed at generalizing as much as possible (and plausible) away from the training data. We have discussed that such generalization results in large numbers of ambiguities and thus large lattices. In fact, in the previous section, we reported that the learned grammar could be run maximally with a lengthlimit of 6, and that beyond this lengthlimit the run-time system could not produce the lattices. This indicates that the algorithms as presented in this thesis were in fact optimized for learning from a very small dataset.

In this section, we present a very brief exploration of the behavior of the system when more training data is added. We consider three additional training sets: the first, which is training data that from the structural corpora described in (Probst & Lavie, 2004). We describe there three versions of the same structural corpus that differ only in lexical choices, but not in the structural make-up of their training examples. Three translators translated parts of these three corpora: one translated version 1, one translated version 1 and version 2, and one translated parts of version 3. For this reason, there is partial overlap between the bilingual data from different translators. Overall, the combined translations from these versions contain 614 training examples. We will refer to this as the 'OtherStruct' corpus. We further added training data that stems from an older, and reduced, version of the functional corpus. Parts of this corpus were translated and word-aligned by three informants, resulting in a total of 806 sentences and phrases. This corpus will be referred to as the 'Reduced' corpus. Finally, we consider the addition of the copula corpus that was used for the copula case study in chapter 8, consisting of 319 sentences. This corpus was translated by one informant. We call this corpus the 'Copula' corpus. In order to simplify our notation, we refer to the standard 120 sentences and phrases as the 'Struct' corpus.

With these corpora at hand, we consider the combinations
Struct+OtherStruct, Struct+OtherStruct+Reduced, and
Struct+OtherStruct+Reduced+Copula, i.e. we add one corpus after an-

other to the training data.

For these corpora combinations, we first learned grammars with the standard settings of all three major learning phases, Seed Generation, Compositionality, and Constraint Learning. Not unexpectedly, even for the combination Struct+OtherStruct we were not able to produce a lattice for a lengthlimit of 6, which was the default lengthlimit in previous experiments.

There are several ways to overcome this problem. We report here on several possible directions, and conclude which directions are most promising for future work. As the topic of this thesis is to learn under a very miserly data scenario, we merely aimed at applying our algorithms to a different data scenario, not at developing new algorithms for situations when more data is available. This should be a topic for future work.

9.8.3 Non-Compositional Grammars for Larger Datasets

When learning compositional rules from a larger dataset, the run-time system is not able to produce lattices for the learned grammars. For this reason, the best strategy would be to limit the generality of the learned rules. This can be accomplished by running the rule learner in a mode where it applies only the Seed Generation algorithm, not the Compositionality or Constraint Learning algorithms. Under this setting, the generality of the grammar is much reduced, as the learned rules cannot combine with each other. They do however capture not simply the observed training examples, but generalize away from the training data to the POS level wherever appropriate.

We learned grammars with Seed Generation only for all combinations mentioned above, and ran experiments for test set 2. The results can be seen in Tables 9.25 and 9.26.

The above results show that the learning system scales to larger training sets, and that translation quality increases as more data becomes available. The copula corpus contains only on the order of a dozen different structures, so that the translation score did not increase with the addition of the copula corpus. It can also be observed that the translation score for the larger training sets exceeds the score that was observed for the compositional grammar that was learned from the original Structural Corpus. This means that with the addition of more training data, improved system performance can be achieved. Further, we observe that more training data results in better performance when comparing grammars that were learned with Seed Generation only. When the training data size was increased, the translation quality of the Seed Generation grammar that was learned from the original Structural Corpus is exceeded.

	BLEU	ModBLEU	METEOR
No grammar	0.0565	0.1362	0.3019
Manual	0.0817	0.1546	0.3241
Struct(SeedGen)	0.0741	0.1498	0.3239
Struct(Compos)	0.0772	0.1519	0.3297
S+O	0.0832	0.1554	0.3263
S+O+R	0.0837	0.1567	0.3308
S+O+R+C	0.0837	0.1567	0.3308

Table 9.25: Results for corpus combinations for Seed Generation only on test set 2. The combinations are abbreviated as follows: "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

Comparison	BLEU	ModBLEU	METEOR
S+O	[0.0488,-0.0058]	[-0.0330,-0.0080]	p=0.102
S+O+R	[0.0474,-0.0073]	[-0.0323,-0.0092]	p=0.0501
S+O+R+C	[-0.0476,-0.0084]	[-0.0324,-0.0093]	p=0.0501

Table 9.26: Corpus combinations for Seed Generation only: confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline on test set 2. The combinations are abbreviated as follows: "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

The above results allow us to plot the translation performance of the learned rules as a function of the training corpus size. In the figures below, we plot the BLEU (Figure 9.6), ModBLEU (Figure 9.7), and METEOR scores (Figure 9.8) for the compared corpora on test set 2.

For comparison, we also report results for the corpus combination Struct+ Reduced. This allows us gain insight into whether the amount of training data or the quality of the structural corpus resulted in the increase in performance. The results in Table 9.27 (p-value and confidence intervals in Table 9.28) indeed indicate that, at least according to BLEU and ModBLEU, a bigger gain in performance results from the addition of the OtherStruct corpus than from the addition of the Reduced corpus. It can be conjectured that this is caused by the higher level of structural variety in the "Other

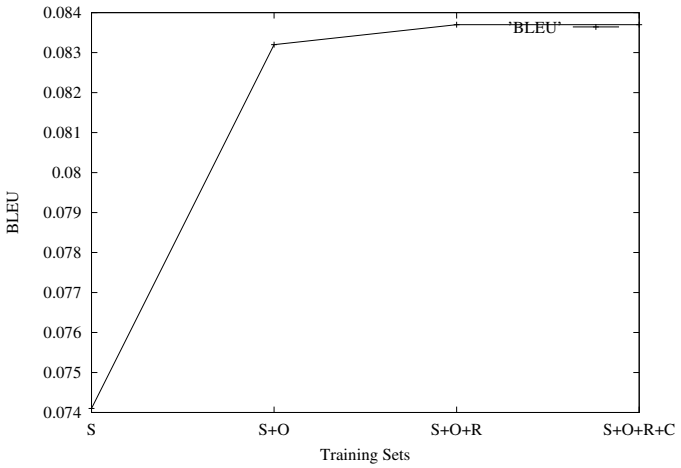

Figure 9.6: BLEU scores for different training corpus combinations on test set 1. The x-axis labels are as follows: "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

Structural" corpus.

In addition to the fact that translation quality increases with larger datasets, a number of important observations can be made based on the observed results.

1. **Compositionality is most appropriate for small datasets.** Compositionality overcomes training data sparseness by generalizing maximally over the training data. This is not only useful, but in fact indispensable when learning from a small dataset. As this thesis focused on such miserly data scenarios, we have consistently found the Compositionality approach to be very important. When more training data is given, Compositionality leads to very general grammars, where the rules combine freely with each other, resulting in extremely, and often infeasibly, large lattices.

2. **Seed Generation captures important structural transfers.** As has been noted before, Seed Generation produces much less general rules, as the rules are flat and cannot combine with each other. This does not mean, however, that Seed Generation does not capture important structural transfers between the languages. As more data

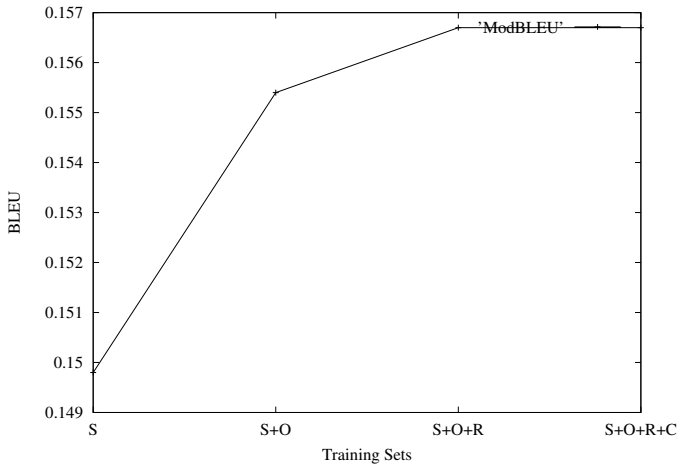

Figure 9.7: Modified BLEU scores for different training corpus combinations on test set 1. The x-axis labels are as follows: "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

becomes available, Seed Generation captures more and more of the structural transfers that are necessary for translations.

3. **A trade-off exists between Seed Generation and Compositionality. The trade-off is a function of the training data size.** Compositionality is a means to overcome training data sparseness. It generalizes to the constituent level, so that combinations of constituent make-ups that were not observed in the training data can be captured by the rules. As more training data becomes available, fewer combinations of constituent make-ups are *not* observed, so that Seed Generation becomes more powerful and the gain from Compositionality becomes less pronounced. Also, as more data becomes available, Compositionality leads to overly general rule that over-combine, leading to infeasibly large lattices. In other words, full Compositionality, as proposed in this thesis, is most appropriate when the training corpora are small.

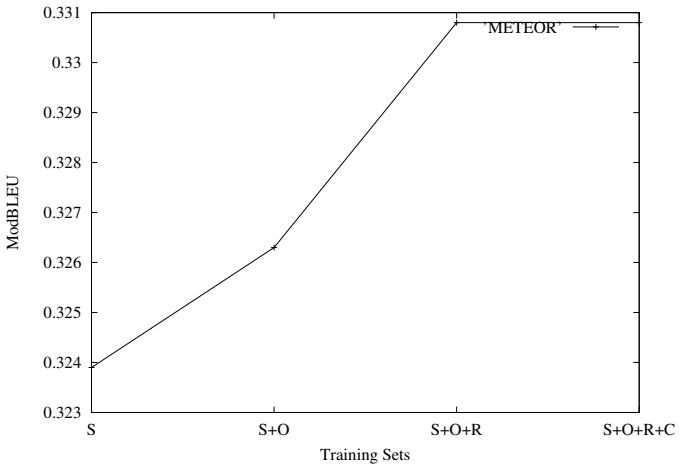

Figure 9.8: METEOR scores for different training corpus combinations on test set 1. The x-axis labels are as follows: "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

9.8.4 Mixed Compositional and Non-Compositional Grammars for Larger Datasets

Although Seed Generation can capture many important structural transfers as more training data becomes available, it should be possible to exploit the strengths of Seed Generation and Compositionality at the same time. For this reason, we have run several exploratory experiments in this direction.

In these experiments, we wanted to explore the possibility of more selective compositionality. In other words, we want to generalize only some, not all rules, to compositional rules. As the original Structural Corpus was designed specifically to capture the most important structures, it is an obvious choice to allow only the rules derived from the Structural Corpus to generalize to compositional rules, and to produce only flat seed rules for the remaining training data. A lengthlimit of 6 was again infeasible, so that the experiments had to be run with a lengthlimit of only 4. The results can be seen in Table 9.29. It was observed that again the addition of more training data results in increased scores. However, the scores are not as high as the ones observed in the original grammars. It can be conjectured that this is mostly due to the forced smaller lengthlimit.

In a similar experiment, we took the top ranking 25% of the composi-

	BLEU	ModBLEU	METEOR
No grammar	0.0565	0.1362	0.3019
Manual	0.0817	0.1546	0.3241
Struct(SeedGen)	0.0741	0.1498	0.3239
Struct(Compos)	0.0772	0.1519	0.3297
S+O	0.0832	0.1554	0.3263
S+R	0.0746	0.1507	0.3297
S+O+R	0.0837	0.1567	0.3308
S+O+R+C	0.0837	0.1567	0.3308

Table 9.27: Results for corpus combinations for Seed Generation only on test set 2. The combinations are abbreviated as follows: "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

Comparison	BLEU	ModBLEU	METEOR
S+R	[-0.0348,-0.0025]	[-0.0231,-0.0064]	p=0.0503

Table 9.28: Corpus combinations for Seed Generation only: confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline on test set 2. The combinations are abbreviated as follows: "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

tional rules from the original grammar. The rules were the top-scoring rules from the pruning test (cf. section 9.5). These rules were combined with the full Seed Generation only learned grammar for the respective datasets. This is justified by the high scores of the Seed Generation only grammars. Adding some compositional rules could lead to additional translation power. The results can be seen in Table 9.30. While it can be seen that the addition of some compositional rules still hurts performance over the Seed Generation only grammars according to ModBLEU and METEOR, this experiment yielded better results than the previous one. Again, the addition of training data improved translation power to a certain extent. This leads us to conclude that such a mixture approach is in fact a promising direction. Future work will need to address the issue of run-time efficiency, so that the learned grammars can actually be used with a higher lengthlimit.

	BLEU	ModBLEU	METEOR
No Grammar	0.0565	0.1362	0.3019
Manual	0.0817	0.1546	0.3241
Struct(SeedGen)	0.0741	0.1498	0.3239
Struct(Compos)	0.0772	0.1519	0.3297
S+O,part.compos(1),LL4	0.0747	0.1474	0.3153
S+O+R,part.compos(1),LL4	0.0749	0.1486	0.3201
S+O+R+C,part.compos(1),LL4	0.0749	0.1486	0.3201

Table 9.29: Comparison of no grammar and several learned grammars for different training corpus sizes. We compare the original Seed Generation only and compositional grammar to grammars of selective compositionality. For the combinations of the Struct corpus with other corpora, only rule derived from the Struct corpus are generalized to compositionality as appropriate. The remaining rules remain flat seed rules. The lengthlimit for the combinations grammars is 4. "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

	BLEU	ModBLEU	METEOR
No Grammar	0.0565	0.1362	0.3019
Manual	0.0817	0.1546	0.3241
Struct(SeedGen)	0.0741	0.1498	0.3239
Struct(Compos)	0.0772	0.1519	0.3297
S+O,part.compos(2),LL4	0.0758	0.1484	0.3173
S+O+R,part.compos(2),LL4	0.0758	0.1493	0.3224
S+O+R+C,part.compos(2),LL4	0.0758	0.1493	0.3224

Table 9.30: Comparison of no grammar and several learned grammars for different training corpus sizes. We compare the original Seed Generation only and compositional grammar to grammars of selective compositionality. For the combinations of the Struct corpus with other corpora, only the top 25% ranking rules from the Struct corpus were included. They were combined with Seed Rules from the entire training corpus. The lengthlimit for the combinations grammars is 4. "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

Another way to address the run-time efficiency issue is to prune the grammars as was described in section 9.5. While this seems a straightforward approach, the run-time system is again the limiting factor. This is because the first step necessary for pruning is to produce a lattice on the training set that a grammar was derived from. This again presented problems of efficiency. We resorted to several helpful heuristics to make this approach feasible:

1. Manual pruning of the OtherStruct training set to eliminate structures that will generalize to undesirable compositional rules. We manually eliminated 201 training examples (leaving this corpus with 413 bilingual training examples). For example, a training example for the structure ADVP→ADJ (i.e. an ADVP consisting only of an ADJ) was eliminated.

2. When generating the full lattice on the training set, a time-out was installed: whenever the transfer engine could not produce a full lattice on a sentence within a certain amount of time, no lattice was produced for this training sentence. The timeout was set to 5 minutes.

3. When pruning the resulting grammars, we chose to prune the grammar to a similar size as we had observed before with high scores. We chose the size of the grammar that was pruned to 75% from the default "original" grammar, and yielded high translation power, which contains 84 rules. However, simply pruning to 84 rules is not desirable, as when pruning from very large grammars, it is unlikely that 'basic', but necessary rules, will be in the top 84 rules. For this reason, we first chose to include three basic rules in the pruned grammar:

```
ADVP::ADVP [ADV] -> [ADV]
(
(X1::Y1)
(X0 = X1)
(Y0 = Y1)
)

NP::NP [N] -> [N]
(
(X1::Y1)
(X0 = X1)
((Y1 NUM) = (X1 NUM))
```

```
(Y0 = Y1)
)

ADJP::ADJP [ADJ] -> [ADJ]
(
(X1::Y1)
(X0 = X1)
(Y0 = Y1)
)
```

The remaining 81 rules were chosen to be the top-ranking rules according to their rule score.

The results of these experiments can be seen in Table 9.31 below. It can be seen that the series of compromises that had to be made in order to overcome run-time limitations yielded limited performance.

	BLEU	**ModBLEU**	**METEOR**
No Grammar	0.0565	0.1362	0.3019
Manual	0.0817	0.1546	0.3241
Struct(SeedGen)	0.0741	0.1498	0.3239
Struct(Compos)	0.0772	0.1519	0.3297
S+O,compos.pruned,LL4	0.0607	0.1386	0.3051
S+O,compos.pruned,LL5	0.063	0.1414	0.3103
S+O,compos.pruned,LL6	0.0636	0.1421	0.3101

Table 9.31: Automatic evaluation results for a grammar learned from the original structural corpus plus the other structural corpus. The learned grammar was 1) scored, and 2) pruned to 84 rules, with c) basic rules such as NP→N included. We report here results for different lengthlimits. 'LL' stands for lengthlimit. "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

Finally, we chose to perform an oracle-like experiment where we modified the grammar of 983 seed rules by adding a few compositional rules (12 rules were added) and deleting some rules that are subsequently subsumed by the 12 new rules (75 rules were deleted). For example, we added the following rules (with constraints):

```
S::S [NP V "AT" NP] -> [NP V NP]
(
(X1::Y1)
(X2::Y2)
(X4::Y3)
((Y3 DEF) = +)
)

{NP,978}
NP::NP [N] -> ["A" N]
(
(X1::Y2)
((Y0 DEF) = -)
)

{NP,979}
NP::NP [N] -> [N]
(
(X1::Y1)
((Y0 DEF) = -)
)
```

These additions and deletions simulate partial and selective compositionality. The results in table 9.32 show that this selective compositionality can in fact result in improved performance.

In summary, when moving to larger datasets, the following two issues must be addressed with the highest priority:

1. **Run-time system efficiency.** The current algorithms yield rules that cannot be applied at run-time under optimal lengthlimits.

2. **Selective Compositionality.** A new mixture algorithm must be developed that only generalizes to compositional rules where appropriate. As this was not the goal of this thesis, such an algorithm has not been designed. However, we showed that in the oracle experiment that selective Compositionality can lead to increased performance. In particular, our algorithms were developed to optimally exploit small training sets. When learning from larger corpora, compositionality could be allowed only if considerable evidence is given that this is desirable. In our approach, due to the sparsity of training data, generalization is allowed if very little evidence is observed.

	BLEU	ModBLEU	METEOR
No Grammar	0.0565	0.1362	0.3019
Manual	0.0817	0.1546	0.3241
Struct(SeedGen)	0.0741	0.1498	0.3239
Struct(Compos)	0.0772	0.1519	0.3297
S+O+R+C	0.0837	0.1567	0.3308
S+O+R+C,man.modified, LL4	0.0810	0.1547	0.3315
S+O+R+C,man.modified, LL5	0.0843	0.1580	0.3363

Table 9.32: Automatic evaluation results for a grammar learned from the full combination of training data, and manually modified to simulate selective compositionality. We report here results for lengthlimits 4 and 5. 'LL' stands for lengthlimit. "S" stands for "Structural Corpus", "O" is "Other Structural Corpus", "R" is "Reduced Functional Elicitation Corpus", and "C" is "Copula Corpus".

9.9 Varying Languages - Hindi→English Translation

Although most of the experiments reported in this document are for a Hebrew→English translation system, our goal was to make the algorithms as language-independent as possible. Hebrew→English is a language pair that lends itself strongly to our approach, in particular as the two languages are very different in word order. If two languages are similar in word order, a word-by-word translation can accomplish acceptable translation performance. When many reorderings are necessary, rules such as the ones learned in our system can truly provide benefit for translation.

As a portability test, we chose our existing Hindi→English MT system. Elsewhere (Lavie et al., 2003), we give details on this system, which is also described in this document in chapter 3. In this section, we report results on an experiment that learns rules from the same structural corpus that was used for Hebrew rule learning. For this purpose, we had the Structural Elicitation corpus of 120 sentences and phrases translated into Hindi and word-aligned. For two phrases, two alternative translations were given, resulting in a total set of 122 training examples. This bilingual corpus served as the training corpus for the experiments reported below, unless otherwise noted.

As for Hebrew→English, we use the learned grammar in conjunction

BLEU	0.1091
Modified BLEU	0.199
METEOR	0.3409

Table 9.33: Hindi baseline evaluation on test set of 122 sentences.

with a Hindi→English lexicon, and compare the performance of the resulting system to a lexicon-only system. The test set consists of 122 sentences of newspaper text. In this section, we present the results in somewhat of a more compressed format than for Hebrew, as the purpose of this section is to only briefly explore the portability of our algorithms, not to perform as full a range of experiments as we did in the Hebrew→English case.

In Table 9.33, we first present the Hindi baseline, i.e. lexicon-only evaluation results on this test set. No lengthlimit was set.

9.9.1 Hindi Grammars with Hebrew Default Settings

Our first experiment was to run the system 'as-is', i.e. with all the major learning algorithms, Seed Generation, Compositionality, and Constraint Learning. No language-specific optimization was performed.

The system learned a number of interesting rules under those settings. Similarly to Hebrew, the Hindi training sentences are given in an romanized version. The romanization used for Hindi here is RomanWX, as described in section 3.5.

```
;;SL: jaMgala meM
;;TL: IN THE FOREST
;;C-Structure:(<PP> (PREP in-1)(<NP> (DET the-2)(N forest-3)))
PP::PP [NP POSTP] -> [PREP NP]
(
(X1::Y2)
(X2::Y1)
(X0 = X2)
(Y0 = Y1)
)
```

The rule learner captured in this rule a canonical reordering between Hindi and English. Hindi uses postpositions, while English uses prepositions. This implies that any PP must be reordered as captured in the rule above.

```
;;SL: muSkIla se vakwa para
;;TL: BARELY ON TIME
;;C-Structure:(<PP> (<ADVP> (ADV barely-1))
    (PREP on-2)(<NP> (N time-3)))
PP::PP [ADVP NP POSTP] -> [ADVP PREP NP]
(
(X1::Y1)
(X2::Y3)
(X3::Y2)
(X0 = X3)
(Y0 = Y2)
)

;;SL:bagIce meM se
;;TL:IN FROM THE GARDEN
;;CStructure:(<PP> (PREP in-1)(<PP> (PREP from-2)
    (<NP> (DET the-3)(N garden-4))))
PP::PP [NP POSTP POSTP] -> [PREP PREP NP]
(
(X1::Y3)
(X2::Y1)
(X3::Y2)
(X0 = X2)
(Y0 = Y1)
)
```

This is a special case of the first Hindi rule presented above. Both postpositions are reordered to appear before the NP. The order of the postpositions is flipped during translation.

```
;;SL: purAnI strIta gAdZI
;;TL: THE OLD STREET CAR
;;C-Structure:(<NP> (DET the-1)(<ADJP> (ADJ old-2))
    (N street-3)(N car-4))
NP::NP [ADJP N N] -> ["THE" ADJP N N]
(
(X1::Y2)
(X2::Y3)
(X3::Y4)
(X0 = X3)
```

```
(Y0 = Y4)
((Y3 NUM) = (Y4 NUM))
((Y3 PERS) = (Y4 PERS))
)
```

This rule captures the fact that Hindi does not use determiners before NPs. The challenge for any translation system into English is then to determine whether to insert 'THE', 'A', or no determiner at all. This rule captures one of the possibilities, indefinite NPs with two modifying ADJPs.

While the above rules capture important generalizations, one observation about the Hindi system is that a large number of rules contain lexical items, which reduces their generalization power. Consider for example the following rule:

```
;;SL:vakwA ne Age kahA
;;TL:ADDED THE SPEAKER
;;CStructure:(<SINV> (V added-1)(<NP> (DET the-2)(N speaker-3)))
SINV::SINV [NP "ne" "Age" "kahanA"] -> ["ADDED" NP]
(
(X1::Y2)
;--;(X3::Y1)
;--;(X4::Y1)
(Y0 = Y1)
)
```

This rule exemplifies a design choice in the learning algorithms: not one-one aligned words generally remain lexicalized with few exceptions. This is done as a precaution against overgeneralization to the POS level. However, in cases such as the rule above, the result is a lexicalized rule that will apply to few if any test sentences at run-time. Future work will address this problem by relaxing the one-one restriction for generalization. In our experiments, it has proven to be a useful safeguard.

Table 9.34 shows the results for the complete system with all major learning algorithms, i.e. Seed Generation, Compositionality, and Constraint Learning, for different lengthlimits. Table 9.34 reports the p-value and confidence interval for the highest-scoring lengthlimit (lengthlimit 1). Note that for all Hindi test, statistical significance is much easier to accomplish because the test set is larger than both Hebrew test sets.

A very unexpected observation can be made: the system performs best with a lengthlimit of 1. This means that the only grammar rules that are

Lengthlimit	1	2	3	4	5
BLEU	0.1142	0.1112	0.1117	0.1103	0.1103
Modified BLEU	0.2076	0.2073	0.2077	0.2069	0.2069
METEOR	0.3531	0.3537	0.3538	0.3539	0.3539

Table 9.34: Translation results and improvements over baseline for learned grammars with Seed Generation, Compositionality, and Constraints.

Comparison	BLEU	ModBLEU	METEOR
Struct,LL1	[-0.0059,0.0005]	[-0.0077,0.0006]	p=0.003

Table 9.35: Grammars learned from Structural Corpus with Seed Generation, Compositionality, and Constraints evaluated at lengthlimit 1: confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline. 'LL' indicates the lengthlimit.

applied are ones whose SL component sequence is of length 1. Closer inspection found that the single most powerful rule in the Hindi→English system is the following:

```
NP::NP [N] -> ["THE" N]
(
(X1::Y2)
)
```

This rule, while seemingly very basic, is very powerful for translation from Hindi into English, because Hindi does not use determiners in noun phrases, unlike English. Therefore, the simple insertion of a definite determiner when translating from Hindi into English results in a gain in translation power.

9.9.2 Non-Compositional Rules for Hindi→English translation

To test whether different learning settings lend themselves better to the Hindi→English translation system, we learned rules using only the Seed Generation module. This results in rules that are flat and do not combine with each other. However, the rules are, wherever possible, generalized to the POS level, so that they capture important structural differences between the two languages. A few examples of learned rules are given below.

```
;;SL:nayIM SAsakIya imAraweM
;;TL:NEW PUBLIC BUILDINGS
;;CStructure:(<NP> (<ADJP> (ADJ new-1))
    (<ADJP> (ADJ public-2))(N buildings-3))
NP::NP [ADJ ADJ N] -> [ADJ ADJ N]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
)
```

This rule exemplifies how Seed Generation results a flattening of the input structure. In this case, the English ADJPs are each represented only by the POS label 'ADJ'. This results in a less general rule that will only apply to the specific POS sequence in the SL component sequence. A similar observation can be made for the following rule:

```
;;SL:bahuwa sIXA Ora sakrIya
;;TL:HIGHLY DIRECT AND ACTIVE
;;CStructure:(<ADJP> (<ADVP> (ADV highly-1))
    (<ADJP> (ADJ direct-2)(CONJ and-3)(ADJ active-4)))
ADJP::ADJP [ADV ADJ CONJ ADJ] -> [ADV ADJ CONJ ADJ]
(
(X1::Y1)
(X2::Y2)
(X3::Y3)
(X4::Y4)
)
```

Here, the ADVP 'DIRECT AND ACTIVE' is not recognized as such, and generalization is only done up to the POS level.

```
;;SL:samAcAra se bahuwa prasanna
;;TL:VERY HAPPY ABOUT THE NEWS
;;CStructure:(<ADJP> (<ADVP> (ADV very-1))(ADJ happy-2)
    (<PP> (PREP about-3) (<NP> (DET the-4)(N news-5))))
ADJP::ADJP [N POSTP ADV ADJ] -> [ADV ADJ PREP "THE" N]
(
(X1::Y5)
(X2::Y3)
```

```
(X3::Y1)
(X4::Y2)
)
```

As in the compositional case, the major problem with the grammar is that a large number of rules contains lexical items, in particular open-class lexical items. Consider, for example, the following rule:

```
;;SL:acCI samAjIka prawimA
;;TL:A GOOD PUBLIC IMAGE
;;CStructure:(<NP> (DET a-1)(<ADJP> (ADJ good-2))
    (<ADJP> (ADJ public-3))(N image-4))
NP::NP ["acCI" ADJ N] -> ["A" "GOOD" ADJ N]
(
;--;(X1::Y2)
(X2::Y3)
(X3::Y4)
)
```

Why would the pair 'acCI'→'GOOD' remain lexicalized, even though the words are aligned one-one? The Seed Generation algorithm contains a number of precautions against generalizing to the POS level. In this case, 'acCI' and 'GOOD' remain lexicalized because the Hindi→English lexicon contains an entry for this pair whose English POS label does not match the POS label given in the parse. While this is simply noise in the lexicon, the automatic safeguard cannot distinguish between noise and an actual case where the POS label cannot be assigned reliably to the Hindi word during rule learning. For this reason, the words remained lexicalized. This points again to the dependence of the rule learner on the reliability of the underlying resources. When the underlying resources contain noise, the rule learner cannot operate under optimal conditions. In the future, such problems can be tackled only with more data, which would result in more evidence. Given more evidence, better decisions can be made. This thesis focused on learning from extremely small datasets. As has been mentioned throughout, the small size of the training corpus implies a poverty of evidence, and thus greater reliance on the underlying resources.

The automatic evaluation results for this experiment can be found in Table 9.36, the corresponding p-values and confidence intervals in Table 9.37 for the highest-performing lengthlimit. As was seen in other experiments throughout this document, the Seed Generation algorithm is able to capture

Lengthlimit	1	2	3	4	5
BLEU	0.1142	0.1148	0.1148	0.1134	0.1134
Modified BLEU	0.2076	0.209	0.2087	0.208	0.208
METEOR	0.3531	0.3557	0.3555	0.3555	0.3555

Table 9.36: Translation results and improvements over baseline for learned grammars with Seed Generation only.

Comparison	BLEU	ModBLEU	METEOR
Struct,SeedGen,LL2	[-0.0064,0.0005]	[-0.0087,-0.0011]	p=0.002

Table 9.37: Grammar learned from Structural Corpus with Seed Generation only, evaluated at lengthlimit 2: confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline.

important generalizations, and can in fact result in improved translation power. In the Hindi case, the Seed Generation only algorithm actually results in better performance than the rules that were learned with all major learning algorithms.

Note that in this case, the best performance occurs at a lengthlimit of 2 (rather than 1, as before). This has to do with the fact that the Seed Generation only grammar contains the following rule, in addition to the basic 'THE' insertion rule above:

```
;;SL:jaMgala meM
;;TL:IN THE FOREST
;;CStructure:(<PP> (PREP in-1)(<NP> (DET the-2)(N forest-3)))
PP::PP [N POSTP] -> [PREP "THE" N]
(
(X1::Y3)
(X2::Y1)
)
```

This rule encodes a quintessential reordering between Hindi and English, where a Hindi postposition (which stands after the noun) turns into a preposition in front of a noun in English. In addition, this rule encodes the insertion of the definite article 'THE' when translating into English. The question now arises why this rule did not benefit to the same extent the

compositional grammar reported above. For comparison, the compositional version of this rule (as learned by the rule learner with Compositionality), is:

```
;;SL:jaMgala meM
;;TL:IN THE FOREST
;;CStructure:(<PP> (PREP in-1)(<NP> (DET the-2)(N forest-3)))
PP::PP [NP POSTP] -> [PREP NP]
(
(X1::Y2)
(X2::Y1)
(X0 = X2)
(Y0 = Y1)
)
```

In this case, the rule is more general and encodes the reordering of Hindi postpositions to the beginning of a NP when translating into English. It would be expected that this rule would result in improved translation. The problem that was encountered in our experiments was that it is often difficult to correctly propose the span of a NP, so that the postposition can be correctly reordered. Hindi is highly ambiguous with respect to the span of NPs. In other words, there are often many sequences of words which could together form a NP. At run-time, this results in a very large number of ambiguities, thus producing a lattice that proposes many possible positions for the postposition. In many cases, the NP however spans only one noun. In this case, the non-compositional rule can correctly reorder the postposition (and insert the determiner). In other words, the non-compositional rule can apply correctly to most cases where this rule should apply. In addition, the compositional rule introduces a large number of ambiguities, most of which are naturally incorrect. For this reason, the non-compositional rule performs better overall.

9.9.3 Learning from Additional Data

We further performed an experiment in which we added additional training data, 149 sentences and phrases from the Reduced training corpus that was also used for Hebrew. The additional data resulted in a training corpus of a total of 271 sentences and phrases. The results, as listed in Table 9.38 suggest that adding more training data has a certain, but small, positive impact on the results. The results are highly statistically significant for

Lengthlimit	1	2	3	4	5
BLEU	0.114	0.1144	0.1144	0.1131	0.1131
Modified BLEU	0.2072	0.2081	0.2087	0.2079	0.2079
METEOR	0.3545	0.3569	0.3573	0.3573	0.3573

Table 9.38: Translation results and improvements over baseline for learned grammars with Seed Generation only, learned with additional data.

ModBLEU and METEOR, and further statistically significant with close to 95% confidence for BLEU.

Comparison	BLEU	ModBLEU	METEOR
Struct,LL1	[-0.0059,0.0005]	[-0.0077,0.0006]	p=0.003
Struct,SeedGen,LL2	[-0.0064,0.0005]	[-0.0087,-0.0011]	p=0.002
Struct+add'l,SeedGen,LL3	[-0.0059,0.0010]	[-0.0088,-0.0008]	p=0.005

Table 9.39: Grammars learned from Structural Corpus and from Structural Corpus with additional training data with Seed Generation only, evaluated at lengthlimits 2 and 3, respectively: confidence intervals (for BLEU and ModBLEU) and one-tailed t-test for comparison between learned grammar and baseline. 'LL' indicates the lengthlimit.

9.9.4 Conclusion of Hindi→English Portability Test

The accomplished improvements for Hindi are plotted in Figure 9.9.

In conclusion, the following inherent weaknesses of the learning algorithms most contributed to the performance drop when porting the system to a new language pair:

- **Verb phrases and verb complexes.** It has been observed that many of the translation variation between Hindi and English lies in verb phrases (Dorr et al., 2003). Verb phrases are not currently captured by our current system. The system could still learn rules for verb complexes, which would be less general than the VP level. However, since it was not the goal of this work to target verb phrases or verb complexes, the structural training corpora do not contain examples for important verb complexes. Possible extensions to the current system to include verb phrase handling are discussed in section 10.3.

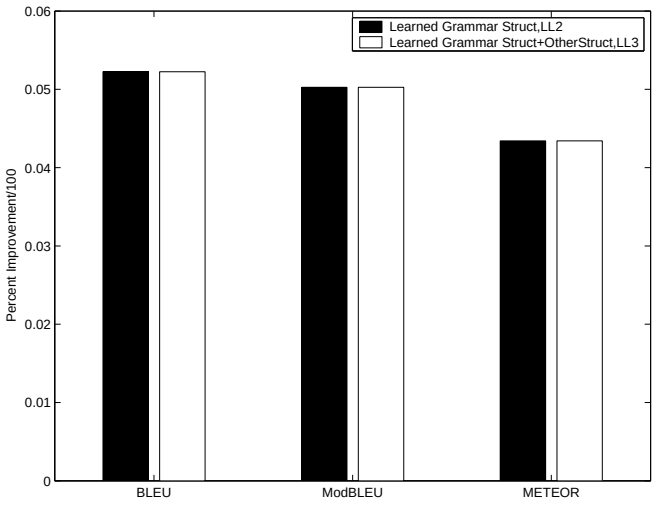

Figure 9.9: Improvement over Baseline for Hindi→English translation.

- **Less reordering during translation.** Our system is strong at discovering how constituents and words must be reordered during translation, so that significant improvements can be gained when such reorderings are actually necessary during translation. When translating from Hindi into English, the most common reordering (outside of verb complexes) is the reordering of a Hindi postposition as a preposition in English. Hebrew→English translation, on the other hand, has 1) more and 2) more frequent such reorderings, such as adjectives that appear mostly after the modified noun in Hebrew and before the modified noun in English.

- **Fewer one-one aligned words.** Our Seed Generation algorithm generally does not generalize to the POS level if words are not one-one aligned. This has been found to be an effective safeguard against overgeneralization. When learning Hindi→English translation rules, many words remain lexicalized because of this restriction. Among other issues, future work (as described in section 10.3) will focus on an extension to handling not one-one aligned words.

- **Weaker lexicon.** A related issue is the strength of the translation lexicon in terms of POS assignments. As we discussed when describing the Seed Generation algorithm (see chapter 6), generalization to the

POS level depends in part on whether the translation pair is marked with the expected POS labels in the dictionary. If the underlying translation dictionary contains fewer correct translation pairs and/or fewer correct POS labels for the translation pairs, less generalization to the POS level is performed during Seed Generation.

- **Ambiguities in determining NP span.** As was said before, one of the most prominent reordering phenomenon for Hindi→English translation is the translation of Hindi postpositions as English prepositions. In order to do this correctly, it is necessary to determine the exact span of the NP, so that the postposition can be inserted into the correct position when translating into English. Because of ambiguities, the English preposition is not inserted in the correct position reliably. This problem can only be solved if all aspects of the learning and runtime system (i.e., morphology, lexicon, rule learning) become stronger, or if the transfer engine allows for probabilistic transfer.

Future work should address these points in particular. This is described in section 10.3. Despite the weaknesses of the current Hindi→English system, it is encouraging to note that without a considerable amount of language-specific engineering, the system can still accomplish a considerable improvement in performance over the baseline.

Chapter 10

Conclusion

10.1 Contributions

Throughout the document, we have made arguments of why this work is an important contribution to the Machine Translation community. The goal of this section is to collect these arguments, at the end of the document, in order to create a more coherent and full picture of how this thesis contributes to MT research. The main contributions are summarized in Figure 10.1.

We have proposed a novel approach to learning transfer rules from bilingual data, using a variety of resources. A series of three algorithms analyzes the training data and infers transfer rules. We showed improved translation quality on Hebrew→English and Hindi→English translation tasks.

In our work, we have focused on learning from extremely small datasets. Most of the grammars presented in this document were learned from 120 training examples. This is a novel contribution, as it has as yet not received attention in the literature. The approach we have taken in this work is to produce rules that are similar in format to what a human grammar writer would write, and that aim at capturing similar phenomena. Manually written grammars are complex in their make-up: they are generally compositional, and they are often annotated with unification constraints. The challenges of this work were 1) to learn grammars in a similarly complex hypothesis space, 2) to do so from extremely limited datasets, and 3) in the absence of a TL parser. We believe that this work has pushed the state of the art research into these directions. To our knowledge, no existing system has approached this particular problem before.

Not only were we able to learn meaningful grammars, however. We have also shown that the automatically learned rules are useful in producing more

Research Contributions of the Presented Thesis

1. Framework for learning transfer rules from bilingual data
2. Improvement of the quality of MT output: translation quality of system with learned rules that is
   - superior to a system that does not use the learned rules
   - comparable to the performance using a small manual grammar written by an expert
3. Proposing a method to overcome the bottleneck for transfer-based MT
4. Encouraging MT research in the direction of incorporating syntax into statistics-based systems
5. Addressing limited-data scenarios with 'frugal' techniques
6. Combining a set of different knowledge sources in a meaningful way, and doing so quickly
7. Human-readable rules that can be improved by an expert or system
8. Rule base input to feedback-based automatic refinement system

Figure 10.1: Summary of contributions.

grammatical output. Our experiments show that the learned grammars capture important linguistic phenomena, in particular reorderings of words and entire constituents and passing of features from source to target language. This shows that we can, to a certain extent, automatically capture the syntax of a language in an unsupervised manner by inferring information from a major language parse and bilingual data. We showed that the automatically learned rules improve translation output, and can yield comparable performance to a small manually written grammar.

Another contribution of this thesis is that it overcomes the problem of long development times that has traditionally been the biggest bottleneck of syntax-based MT. In the past, most transfer-based systems relied on an extensive manually written rule base, which often took many human years to develop. This approach is targeted specifically at overcoming this bottleneck by introducing an automatic way to infer such transfer rules. Furthermore, the rules are learned from very small training sets. This allows us to minimize the amount of necessary human involvement.

One of the most important contribution of this work is to encourage the Machine Translation community to further integrate statistical techniques with syntactic approaches. Syntax-based MT and statistical MT have not traditionally been regarded as approaches that can be integrated easily. Here we propose a framework within which the integration can be accomplished. The learning of syntactic rules is done independently of decoding. Our learning approach allows for a very fine-grained hypothesis space. We can learn rules within this space, and then integrate them easily into the statistical system. The rule learning module requires some word-aligned data, as this work as focused on minority languages, and thus on very small, but high-quality corpora. Future work may address rule learning from any bilingual corpus with automatic word alignments. Then the approach presented here could be used within any statistical system, using whatever training data is available for the statistical system as well.

The lack of large training corpora for the language pairs of relevance for this work forced us to use techniques specific to small high-quality corpora. For instance, we opted for eliciting word alignments in order to reduce noise in the training data. The rule learning algorithms were carefully designed to work on small corpora. For instance, the Compositionality techniques presented in this document are aimed at maximizing the generality and thus applicability of the grammar. More lexicalized rules would apply to much fewer circumstances meaning that more training data will be necessary to get broad coverage. We also showed that Compositionality Learning is optimized for small corpora, and can boost performance under such a scenario.

For larger corpora, we showed that more selective Compositionality is necessary.

When learning rules within a rich hypothesis space and from very little data, one cannot overstate the importance and difficulty of integrating a variety of knowledge sources in a meaningful way. One of the premises of this work has been to integrate whatever knowledge is available. This means making the best use possible of an English parser as well as morphology modules, but also a head table for English, word alignments, an incomplete and not noise-free dictionary, a list of irregular English verbs, etc. The basic difficulty is that none of these resources are perfect or noise free. In fact, they often disagree, and it is difficult to resolve the conflict, as well as any remaining ambiguity, meaningfully. For example, we are often faced with a question such as "What is the most likely part of speech for a given Hebrew word in this context?" We can consult several resources: 1) we can consider the POS of the aligned English word, which we know from the parse. This is a good idea if the Hebrew and English words are aligned one-one. 2) we can consider the Hebrew-English dictionary. In order to do that, we must first determine the most likely root of the Hebrew word. This can in principle be accomplished using the Hebrew morphology module. In practice, however, any Hebrew word could have multiple analyses, and thus multiple possible roots. Once the root is determined, there may be several entries for a given Hebrew word in the dictionary, so that it is not obvious which one is the correct entry. What if there is only one entry, and its POS is different from the POS proposed by the morphology module, and in turn different from the POS of the aligned English word? How can we design criteria that will meaningfully combine these sources of information?

In this thesis, we did not focus on such questions of inconsistency of underlying resources. They are common to any natural language system that incorporates morphological or structural analysis. The fact that despite these difficulties, we succeeded in learning meaningful rules and improving translation output definitely speaks strongly in favor of our approach. If and when the underlying resources improve in quality, the rule learning system will in turn improve in quality as well.

As we do not want to be idle and wait simply until other resources improve, we also designed our learning approach such that the learned grammars can serve as a 'raw version' of a transfer grammar that is refined either by an expert or an interactive rule refinement module. This is because the rules are of very fine granularity, in particular because of the unification constraints. This allows for interactive refinement with user feedback. The system described in (Font-Llitjós, 2004) proposes to use user corrections to

refine learned grammar rules. Non-expert users correct MT output that was produced with the learned rules. The system then assigns blame to specific rules, and fixes the rules so that the MT output will no longer be faulty. The importance of the work presented here is that the rules have enough specific details and constraints to allow for this process to happen.

Finally, one obvious contribution of this work is that it provides a synergy between statistical techniques and transfer-based techniques: as was said above, it allows for the integration of syntactic rules into a statistical system. But in addition, the learned rules are human-readable. This means that experts can post-process the learned rules. In other words, the work developed here can be seen as a way to bootstrap the development of new grammars. This was not the original intention, and is not how we use the rule learning module in our work. However, it is important to stress that our rule learning system can be used also in the context of transfer-based systems where the automatically learned rules serve as hypotheses that human grammar writers can post-process, refine, and augment. With first hypotheses given to the experts, a speed-up in grammar development time can be expected.

10.2 Lessons Learned

After presenting the rule learning algorithms as well as their performance in practice, this chapter will summarize once again the conclusions that can be drawn and the lessons that were learned.

1. When combining a number of resources, **the performance of the algorithm depends to a certain extent on how reliable the underlying resources are**, and how they are combined when they contradict each other. This implies that further development of underlying resources could in the future lead to better rule learning performance.

2. The three learning phases, **Seed Generation, Compositionality, and Constraint Learning perform well under different circumstances**:

    - **Seed Generation** performs well when a large amount of training data is given. Seed Generation can be seen as a way to induce flat transfer rules from data, i.e. rules that are generalized to the POS level wherever possible, but that cannot combine with

each other. Its specific boundary is completely lexicalized rules. This specific boundary is not generally reached, and thus Seed Generation provides a certain amount of generalization over the training data.

- **Compositionality Learning without the assumption of Maximum Compositionality** performs well especially when there is little training data. Under those circumstances, the costly training-time efficiency is not an issue. Furthermore, this algorithm generalizes only when the generalization is licensed by previously learned rules and thus by the training data. Generally, this algorithm tends to perform comparable to Compositionality Learning with the assumption of Maximum Compositionality.
- **Compositionality Learning with the assumption of Maximum Compositionality** is most appropriate when maximum generalization over the training data is desired. This algorithm generally produces rules that are most human-readable and most similar to rules that a human grammar writer would write when faced with the same training example. Compositionality learning with the assumption of Maximum Compositionality has the further advantage of linear learning time. When it is used for large training sets, however, the grammar must be pruned of spurious rules, as the generality of the grammar results in potentially very large lattices at run-time.
- **Constraint learning** allows us to limit the generality of the grammar in a way that is different from pruning: pruning eliminates rules completely, while constraints limit the applicability of a rule or limits the set of possible outputs. It is useful to eliminate arcs from the lattice that would merely slow down the run-time system, and it thus results in improved run-time efficiency without compromising translation quality to a large degree.

3. **Pruning** of rules from the grammar based on the estimated translation power of the rules is another means to improve run-time efficiency. The pruning of ineffective rules results in the elimination of arcs from the lattice that do not provide good partial translations, and that merely slow down the run-time system. Ideally, pruning also eliminates arcs that are bad translations and that would have been chosen by the decoder had they been provided. In such cases, pruning can achieve an improvement in translation quality.

4. **Language Portability depends on the specific language pair**. Our portability test to Hindi→English showed that we can accomplish improved translation power on this language pair, but that the Hindi→English translation task does not lend itself as well to the structural learning task as Hebrew→English does. This is caused, among others, by the following:

   - Hindi→English exhibits fewer reorderings than Hebrew→English. For example, Hebrew→English translation requires reordering of adjectives and nouns, reordering of noun compounds, etc. On the other hand, Hindi→English translation mainly requires reorderings in prepositional phrases, where Hindi postpositions must be moved in front of the NP. These reorderings are less common than the Hebrew→English reorderings, causing the learned rules to have less effect.
   - A large portion of the translation mismatches between Hindi and English lie in VPs, an area of research that this thesis did not tackle. In section 10.3, we will discuss this issue in greater detail.

5. **Additional Training Data can lead to better performance.** We showed that the addition of training data can lead to better performance. Because of limitations of the run-time system, lenient compositionality is not the optimal choice in this context. Instead, producing flat rules that do not combine leads to improved performance. Future work should include devising more selective methods of compositionality. When learning from a very small corpus, as was done throughout this thesis, the best choice is to introduce compositionality to overcome training data limitations wherever possible. When more training data is available, there are fewer limitations with this data, so that compositionality leads to more ambiguity while not leading to much increased generalization power.

What can we learn from these lessons? One conclusion is that the different learning algorithms that were proposed are suitable for different circumstances. The learning system is flexible enough to run in a variety of settings, where the specific settings can, and should, be adapted to the situation at hand. What settings to choose will depend to a large degree on the language pair as well as on the amount of training data available.

We can conclude that we have designed a flexible and novel approach to learning transfer rules for MT. The full suite of learning algorithms is optimized for scenarios where very little bilingual training data is available.

10.3 Future Work

While any thesis is a substantial piece of research work, it also opens up more research directions. This thesis is no exception. Future directions for this work can be seen as falling into two categories: first, pursuing a similar direction in learning rules from very small corpora and for minority languages, and second, modify the algorithms developed here to work on large, uncontrolled corpora.

10.3.1 Learning from Larger Corpora

We have presented a framework for learning transfer rules for Machine Translation. In particular, we have focused on learning from a very small corpus. In the future, we hope in particular to apply the rule learning approach described in this thesis to large uncontrolled corpora. This can safely be considered the largest and most promising area for future work: while we believe that the automated learning techniques developed here will be useful for large corpora, we have not explored this area in detail. Our feasibility study and our study with non-compositional rules showed that more selective compositionality is the most promising avenue in this area. In the future, we hope to further pursue this direction. If we succeed, then we will have an extremely adaptive learning approach that can work successfully for a variety of language pairs and corpus sizes.

The first step in this direction is to develop robust techniques to learn from automatic word alignments and automatic parses. When learning from much larger corpora, it will not be feasible to get reliable manual word alignments. The challenge will then be to use automatic alignments, which can be expected to be noisier than manual alignments.

Further, the two areas that will have to receive most attention in the future are 1) training corpora enhancement, and 2) rule scoring. We will now briefly discuss these areas.

10.3.2 Training corpora enhancement

Training corpora enhancement is a technique that allows one to create training examples at different levels, e.g., NPs, ADJPs. We have developed a technique for accomplishing this task, as described briefly in (Lavie et al., 2003) and (Font-Llitjós et al., 2004). For example, a training example at the sentence level can be split up at different levels, e.g., NPs, PPs, etc. as follows: A new training example can be created by extracting the partial

parse for the NP (or PP), the sub-sentential TL chunk, as well as the corresponding SL chunk, and the relevant word alignments. We did not use this technique in the experiments reported here, because it is not a completely fail-safe operation, and because the structural corpus was designed specifically to contain training examples at different levels. When learning from uncontrolled corpora, however, training data enhancement will be very important, because all training data will be given only in the form of full sentences. The Compositionality module depends on having rules learned at different levels, and at run-time rules can only combine if they are at different levels.

10.3.3 Rule Scoring and Filtering

The second specific component that will have to receive additional attention when learning from larger corpora is rule scoring and filtering. The rule learning system will learn a larger number of rules from a bigger training corpus. The advantage of this will be that frequency scores and other statistics will become usable. However, any training corpus, in particular large uncontrolled corpora, will contain some amount of noise, resulting in noisy rules. Thus more noisy rules will be learned, and filtering will become very important. Overall, however, we believe that adapting the present approaches to work on much larger corpora, and with automatic parses and word alignments, is the most prominent future direction of this work.

10.3.4 Constructions, Divergences

In the section on advanced structural learning, we discussed different approaches that have been proposed to explaining variation that occurs during translation. One theory is the theory of constructions, which views all of language as specific constructions, and argues that syntax derives from how a language chooses to express a certain concept (Levin & Nirenburg, 1994). A different approach, the divergence approach (Dorr, 1992) classifies translation variation into several categories, such as categorial variation or structural divergence. In this section, we will discuss how some of the types of translation variation could be handled by our rule learner, i.e. how our algorithms could be expanded to be able to capture more variation than it is able to capture now.

We will discuss the following types of variation that are both high priority and that hold promise for boosting the strength of the system beyond what it can do so far.

1. **Verb handling.** These two categories require some complex handling of verb phrases or verb complexes.

2. **Categorial variation.** We have already discussed this issue in the context of lexicon enhancement, in particular for Hebrew adverbs. Here, we provide possible extensions.

3. **Structural variation.** Structural variation, e.g., conflation of two words into one, is already handled in part by the existing algorithms. Again, we will discuss areas of possible extension.

4. **Other constructions.** While arguably all of the above fall under constructions, we propose here a more general approach to handling constructions. Such an approach could apply to any type of construction, while the previous three were specific to certain phenomena. Often, constructions provide a certain amount of generalization power, but are usually bound by specific lexical items. We will discuss possible approaches to handling constructions.

Verb Handling

Let us fist turn to the handling of verbs, as it should receive the most attention in future work. In this thesis, we have chosen not to handle VPs. This is because VPs are very diverse, and their make-up often depends on the specific verb's subcategorization information. For example, some verbs require a direct object, some a direct and an indirect object, etc. The subcategorization information is specific to a verb. Our rules, however, capture structures that are generally applicable to all or most words of a given POS. Since we are using a very small training corpus, we aim to not learn rules that are bound by specific lexical items, especially if those lexical items are of an open word class.

How can verb complexes be learned? We will illustrate the issues with examples from Hindi and Hebrew. In Hindi, many verbs appear in the citation form, and the verb form is expressed using an inflected light verb such as 'honA' or 'karanA', which expresses the form that the verb is in. For example:

```
V::V |: ["praxAna" "karawA" "WA"] -> ["used to give"]
(
(X1::Y1)
)
```

and

```
V::V |: ["praxAna" "karawe" "hue"] -> ["giving"]
(
(X1::Y1)
)
```

The inflected form of 'karanA' ('karawA' and 'karawe' in the examples) together with the constant lexical items 'WA' and 'hue' mark the inflection of the verb. In English, the first is expressed as a verb complex, whereas the second is expressed as an inflected form of the verb.

In order to handle such cases, we would first need to design an extensive verb corpus that elicits verbs in all different forms. The corpus would furthermore need to elicit all verb forms for a number of different verbs, because forms are sometimes expressed differently depending on the specific verb. Irregular verbs are one example of this phenomenon. Another example is that in Hindi, some verbs occur not with forms of 'karanA', but with forms of 'honA', as in the example below:

```
V::V |: ["kula" "hoke"] -> ["adding"]
(
(X1::Y1)
)
```

Some verbs occur with neither of these light verbs, but in some other form.

Thus, the first challenge would be to design a corpus that would provide enough training examples for all different verb forms. This is however not enough. Consider again the two examples of 'karanA'. The first could be generalized to the following rule:

```
VC::VC  [V "karawA" "WA"] -> ["used to" V]
(
(X1::Y1)
)
```

This rule is a rule that the current learning system can infer, and that the run-time system can handle. The second example, however, would need to be handled as follows:

```
VC::VC |: [V "karawe" "hue"] -> [V]
(
(X1::Y1)
((Y1 FORM) = GERUND)
)
```

The above two VC rules could be learned in almost the desired form provided that 1) there are examples in the training corpus 2) the Hindi and English morphological analysis modules give the correct analyses and 3) the relevant word translation entries can be found in the dictionary. The problem with this rule is that the run-time system does not currently support morphological generation in English, as all morphological generation information is encoded in the dictionary. The best solution for this problem would in fact be an English generation module.

Finally, the proposed handling of verb complexes does not correctly cover all verbs. Consider the following example for Hebrew→English translation, where the English auxiliary and verb 'spread' across the sentence:

```
ANI TMID    AAHWV               AWTX
I   always  like.1st.sg.fut     you.sg.f.acc
'I will always like you'
```

If we had learned a VC rule that inserts a 'WILL' in English to mark future, it would look as follows:

```
VC::VC [V] -> ["WILL" V]
(
(X1::Y2)
(X1 FORM) = FUT)
)
```

This rule correctly turns a Hebrew future verb into an English future verb complex. It can however not apply to the bilingual sentence pair given above. The only way for it to apply properly would be a rule as follows:

```
S::S [NP ADVP VC  NP] -> [NP VC(Part1) ADVP VC(Part2) NP]
(
(X1::Y1)
(X2::Y3)
(X3::Y2)
(X3::Y4)
```

```
(X4::Y5)
)
```

In other words, the VC rule 'interleaves' with the S rule. This is something that the current run-time system does not support. For this reason, we propose a two-step process, where we first produce the English sequence

```
[NP  VC          ADVP     NP]
'I' 'WILL LIKE' 'ALWAYS' 'YOU'
```

and then, in a post-processing step, reorder the produced sequence to produce the proper English ordering. The post-processing module could be applied to a variety of cases, not only to distributed verb complexes. It will be similar in spirit to what was proposed in (Dorr et al., 2002), which reorders English sequences to more closely reflect the constituent order of another language. This is a potentially large area of future work.

To summarize, we propose to handle verbs and verb complexes with the following steps:

1. Design a training corpus that elicits verb complexes and verbs in all forms.
2. Integrate English morphological generation.
3. Add a post-processing module to deal with interleaving rules.

Categorial Variation

The second type of translation variation that should be handled in future work are categorial variations. We have already proposed a method that expands the lexicon automatically by scanning the training data for examples of systematic categorial variation, for example between English adverbs and their Hebrew expression as a preposition plus a noun. This approach could be expanded to cover a larger number of contexts. For example, in the solution proposed in section 7.6.3, we automatically deduced derivational morphology rules for English. In the future, we will consider using such resources as the categorial variational database (or CatVar for short, (Habash & Dorr, 2003)). Such a database will allow us to find the correspondents for a given English word in a different part of speech, and will thus allow us to detect more cases of categorial variation. For example, if a Hebrew or Hindi word was translated into an English word, but the correspondence is not found in a dictionary, the current algorithm leaves the words lexicalized.

In the future, we could check whether there exists a lexicon entry for all categories of the English word. If such cases could be detected, they could be generalized rather than being left lexicalized.

Structural Variation and Conflations

Structural variation is already handled in the current system. For example, the Hebrew direct definite object marker 'ET' is simply retained as a lexical item in the rule, and is eliminated when translating into English:

```
S::S [NP V "ET" NP] -> [NP V NP]
```

This is one type of structural variation. Similarly, we have proposed a method for handling conflations in section 7.6.2, which works by conflating two words of a compound into one in the learned rules.

In order to expand on what has been done so far, one area of future work is not only two-one alignments, but also many-one alignments. Such cases are often bound by specific lexical items, i.e. they do not apply to all (or most) words of a given word class. For this reason, we propose exploring further our work on dictionary enhancement, rather than learning highly lexicalized transfer rules. Dictionary enhancement, as opposed to capturing phenomena in transfer rules, has the advantage that the transfer rules will not be overly specific.

More generally, it should be noted that this and other algorithms are dependent on examples in the training corpus. When working with a very small training corpus, such phenomena are not always observed. It is particularly hard to design a corpus in one language (without knowledge of the other language) and guarantee that certain phenomena will occur. For verbs, we can design a specific corpus that contains all verb forms. For structural variation, we must restrict ourselves to handling those cases that occurred in the training data. This holds especially for phenomena that are bound by lexical items, such as compounds.

Other Constructions

Constructions constitute a sizeable part of language. Some authors argue that all of language is constructions, but that in some cases languages happen to use the same constructions (Levin & Nirenburg, 1994). From our perspective, constructions can be viewed expressions that are partially bound by specific lexical items, and allow partially for generalization. This is a more general concept than the ones described in previous sections: We propose

here a method for handling constructions in general, while in the previous few sections, we discussed approaches to very specific phenomena, whereas here we frame the problem much more generally.

Consider the following construction in Hebrew and English:

```
ZH  @WB  B  AINIM $LI
it  good in eyes  my
'I like it'
```

Note that in such a construction, any NP can occur in the subject position in English. Similarly, the possessive in Hebrew can take any person and number value.

Constructions provide at least two unique difficulties:

1. **Recognize which parts can be generalized.** This requires a number of training examples for each construction, so that the common and different parts can be determined automatically. As was said above, this is a challenging elicitation task, as it is often hard to predict what constructions exist in different languages.

2. **Capture the actual generalization in rules.** The above example illustrates this difficulty well. In this case, a possessive translates into a NP, which is not currently handled by the rule learner. The learning algorithms should be expanded to handle these cases, but only if enough training examples are observed. It would not be desirable to learn a rule that indeterminately translates a Hebrew possessive into an English NP. Further, the learning system would need to ensure that all the necessary features (in this case, number and person) are transferred properly in translation.

10.3.5 Other Areas for Future Work

Another area of future work will be to integrate the rule learner with an automatic morphology detection module. Unsupervised morphology is an active area of research that is likely to mature over the coming years to a point where it can be used successfully with other unsupervised techniques such as rule learning.

Further, we will consider the combination of the two learning phases of Compositionality and Constraints into one wherever this is appropriate. We have argued above (sections 3.2 and 8) why it is desirable to keep the learning phases separate. There are, however, cases where it could be useful

to combine them. One such example are rules of the form that were described in section 8.6. There, we described pairs of rules that agree in the SL component sequence, but not in the TL component sequence. We aimed at learning value constraints that can distinguish between the two cases, and thus determine the structure of the TL output. What if there are no value constraints that can distinguish between the two? In such cases, it could be useful to drop the level of generalization of the two rules, so that they differ in their SL component sequence, and the ambiguity is eliminated. This could only be accomplished by another feedback loop into the structural learning phase. Those and other cases would provide interesting insight into the interaction between structural and Constraint Learning. One interesting area of future work could be the exploration of such cases.

We believe that the approach presented here and similar approaches will be used in coming years to push MT research beyond its current limitations. Statistical information will play a big role, but deeper analysis techniques should not be overlooked or ignored. We have shown here that syntactic, morphological, and other feature information can be put to effective use. In the future, we hope to continue our work in this direction, and hope that other researchers will follow suit.

Appendix A

Lattice Scoring

In this section, we describe a *recall*-based method to evaluate a lattice before decoding. Assessing the quality of a lattice, rather than of the final translations, allows us factor out the effect of the decoder on the translation score. By doing so, we can gain insight into the performance of the rule set as a whole. We can also use it to measure the quality of specific rules, as was done in section 9.5, because each arc in the lattice is tagged with the rules that were used to create it. The evaluation method works by measuring how many of the n-grams in the reference translations can be found in the lattice. If all of the n-grams in the reference translations are in the lattice, then a perfect decoder would produce a perfect translation.

The lattice of partial translations contains a set of arcs for each sentence. Our evaluation metric scores each arc separately as follows: we consider all n-grams of length up to k (i.e. unigrams, bigrams, trigrams, ... k-grams) in the partial translation and match them against the reference translation. For each n-gram that is found in the reference translation, we add to the n-gram score

$$matchscore_{ngram} = \frac{1}{ngrams_{RefTranslation}},$$

where $ngrams_{RefTranslation}$ is the number of n-grams in the reference translation. This measures how many of the n-grams in the reference translations are actually found in the partial translation. The number of n-grams in the reference translation is given by $ngrams_{RefTranslation} = c - n + 1$, where c is the number of words in the reference translation. For example, a sentence of c =20 words contains 17 4-grams. Each matched 4-gram will add $\frac{1}{17}$ to the arc score. As mentioned earlier, a user-specified parameter k specifies the length of the highest n-gram that is measured. By default, k is set to

4, meaning that the metric evaluates unigrams, bigrams, trigrams, and 4-grams. This is a reasonable choice, as it is often used in automatic n-gram based MT evaluation metrics, such as BLEU (Papineni et al., 1998).

After collecting all match scores for an arc, we must combine the unigram, bigram, ..., k-gram scores into a single score for each arc. This is done by interpolating the scores for each n. The weight is uniformly distributed over all n-gram scores, so that $\lambda = \frac{1}{k}$. The score of an arc i is then determined by:

$$arcscore_i = \sum_{n=1}^{k} \lambda m_n \frac{1}{c-n+1},$$

where c is the number of words in the reference translation, m is the number of n-grams that match the reference translation (for a given n), and k is the length of the highest n-gram considered.

After each arc is tagged with a score, we can obtain two types of information: the first is to assign a score to each individual rule (as used in section 9.5), the other is to assign a score to the entire lattice.

Rule-level scores can be computed because we know what rule or rules were applied to produce a specific arc. Each arc is assigned to the rule which was the top-level rule in the arc production. For example, if an arc was produced by a sentence-level rule $S, 1$, which filled its subject with an NP rule $NP, 2$, then this specific arc is only assigned only to $S, 1$, because it is the top-level rule for this arc production. The scoring mechanism abstracts away from mistakes that are made by lower-level rules also involved in the arc production: for a given span of indices and a given rule, only the highest-scoring arc is assigned to the top-level rule. This eliminates problems caused by lower-level rules as well as ambiguous lexical selection.

For each individual rule, we sum the arc scores for all arcs 1) to which the rule contributed as the top-level rule, and 2) which have the maximum score for their span. We then divide by the number of rule applications in order to get a confidence (precision) score for the rule. This precision score captures the average quality of an arc to which the given rule contributed as the top-level rule. If we denote with k the number of arcs to which rule R_i contributed, and we denote the subset of arcs in the lattice to which rule R_i contributed as $ArcSet_{R_i}$ (i.e. $|ArcSet_{R_i}| = k$), then the rule precision is:

$$RulePrecision = \frac{\sum_{j=1}^{k} ArcScore_j}{k}, j \in ArcSet_{R_i}$$

After each arc is tagged with a score, we can also combine the arc scores

to obtain a score for the lattice as a whole. In doing this, we do not simply add the arc scores. Rather, we discount additional arcs that account for the same part of the test sentence. More specifically, when a new arc is added that 'explains' (i.e. correctly translates) a part of the sentence that has been explained before, then the score for this arc should be highly discounted. This will prevent the scores from increasing by the addition of arcs that essentially do not add anything new to the lattice. We therefore first distribute the arc score over the matched words in it by taking $\frac{arcscore_i}{nummatched_i}$, where $nummatched_i$ is the number of matched words in arc i. We then consider each word j in arc i that matches the reference translation (there are $nummatched_i$ js for a given arc). Each partial score is discounted based on $count_j$, the number of times this part of the reference sentence, j, was already accounted for by existing arcs, including the current one (so that if an arc accounts for j for the first time, its score is not discounted). The final lattice score is then given by:

$$\frac{\sum_{i=1}^{N} \sum_{l=1}^{nummatched_i} \frac{arcscore_i}{nummatched_i} * \frac{1}{2^{count_{j_l}-1}}}{corpussize}$$

For convenience, we repeat here the notation used in this metric. N denotes the number of arcs in the lattice, $nummatched_i$ denotes the number of words matched in the reference translation for a given arc. $count_{j_l}$ denotes the number of times a certain word index, j, for a given sentence was accounted for by arcs in the lattice, and j_l indicates the word index of the lth word in the arc that matched the reference translation. $corpussize$ indicates the number of sentences in the test corpus.

The metric emphasizes recall in the lattice by rewarding those arcs (or parts thereof) that match the reference translation. The longer the match, the higher the reward. Also, it rewards instances where an arc accounts for a part of the reference sentence that was not previously accounted for, while highly discounting arcs (or parts of arcs) that introduce redundancy into the lattice.

Note that the lattice scoring metric as described here is only possible because the transfer engine and the decoder are two separate modules. If this were not the case, then the rule level evaluation method would need to be modified. In such a case, we could assess the quality of individual rules as follows: in order to asses the quality of rule R_i in grammar G, use grammar G to translate a development set, or as here the training set. This results in a translation score for G, $score_G$. Then eliminate R_i from G, resulting in modified grammar $G' = G \setminus \{R_i\}$. Run G' on the same development or

training set to obtain $score_{G'}$. The quality of R_i can then be estimated by $score_G - score_{G'}$. This method would be an effective way of estimating rule quality. Its drawback is efficiency: especially for large grammars, the operation would be expensive, because the training or development set must be translated separately to assess the quality of each rule. For this reason, the lattice scoring metric used here is preferable in practice.

Appendix B

Sample Translations

Out of 119 test suite sentences, 62 were translated differently with a grammar than without. Note that this does not mean that for the remaining 48 sentences, the lattices are identical in both cases. In fact, the lattices are never identical. However, in 71 cases the decoder chose a different best path through the lattice.

No Grammar:	under to mi@h he found border back that it will remember well from childhood his
Learned Grammar:	under to mi@h it finds old book that he will remember well from his childhood
Reference Translation:	Under the bed he found an old book that he remembered vividly from his childhood
No Grammar:	the decision the final by lot his for the captain cpwih become known in $bwa show
Learned Grammar:	the final decision on his fate of the captain cpwih become known in $bwa show
Reference Translation:	The final decision on the captain 's fate is expected will be announced publicly next week
No Grammar:	under cut <UNK> the medicine will respond in her with drugs different the team that if her the party the international most successful
Learned Grammar:	under cut <UNK> the medicine will respond in her with other drugs the team that if her the most international group successful

Reference Translation:	Under certain conditions the drug will react strongly with other drugs
No Grammar:	it past development long and swab
Learned Grammar:	he studied long process and swab
Reference Translation:	He underwent a long complicated procedure
No Grammar:	we protect activities quick and positive
Learned Grammar:	we ensure prompt and safe operations
Reference Translation:	We guarantee fast secure transactions
No Grammar:	the house stood finally long and pwtl
Learned Grammar:	the house stood long finally and pwtl
Reference Translation:	The house stood at the end of a long windy road
No Grammar:	they with iqxw extension short precedent and not mwgblt of known the finish
Learned Grammar:	they with iqxw short extension precedent and not mwgblt of known the finish
Reference Translation:	They asked for an unprecedented unlimited extension of the deadline
No Grammar:	the change the second the major at rule iiw$m more years
Learned Grammar:	the second change the strong in law iiw$m more years
Reference Translation:	The second big change in the law will be introduced in two years
No Grammar:	the radio broadcast news political great every time
Learned Grammar:	the radio broadcast news important political every time
Reference Translation:	The radio broadcasts important political news every hour
No Grammar:	troops religious title $p&h hwmi&w you views their
Learned Grammar:	religious groups in the cost impact hwmi&w you their ideas
Reference Translation:	Influencial religious groups voiced their opinions
No Grammar:	the response can to make her military grave
Learned Grammar:	the response can to make her major military
Reference Translation:	The response could be a serious military attack
No Grammar:	we coat for communication public grand today

Learned Grammar:	we coat for important public announcement today
Reference Translation:	We expect an important public announcement today
No Grammar:	it will wear jackets black and prolonged
Learned Grammar:	he wore black jacket and prolonged
Reference Translation:	He wore a long black coat
No Grammar:	the book the back of the hebrew contains many matters of &ninim
Learned Grammar:	the old book on hebrew contains many matters of &ninim
Reference Translation:	The old Hebrew book contained many interesting stories
No Grammar:	the girl reduction not understood why the hwri her left
Learned Grammar:	the little girl not understood why the hwri her left
Reference Translation:	The small child did not understand why her parents had left
No Grammar:	the house the beautiful near river sell for 100000 dollars
Learned Grammar:	the beautiful home near river sell for 100000 dollars
Reference Translation:	The pretty house near the river was sold for $ 1000000
No Grammar:	the attack the keen on plwgh last five days
Learned Grammar:	the fierce attack on plwgh last five days
Reference Translation:	The intense assault on Falluja lasted for five days
No Grammar:	candidates right choice
Learned Grammar:	successful candidate elected
Reference Translation:	A good candidate was selected
No Grammar:	the students the successful transfer you the examination
Learned Grammar:	the successful students study you the examination
Reference Translation:	The good students passed the exam
No Grammar:	the president be ashamed selected of secretary new
Learned Grammar:	the president be ashamed picked in the new secretary
Reference Translation:	President Bush chose a new Secretary
No Grammar:	the book the old contains stories of &ninim
Learned Grammar:	the old book contains stories of &ninim
Reference Translation:	The old book contained interesting stories

No Grammar:	we coat for communication grand today
Learned Grammar:	we coat for important communication today
Reference Translation:	We expect an important announcement today
No Grammar:	the students the young learn about countries learn
Learned Grammar:	the young students studied for different countries
Reference Translation:	The young students learned about different countries
No Grammar:	students youths on the road all want to learn countries learn
Learned Grammar:	young students with access all wish to study various worlds
Reference Translation:	Young students often want to learn about different countries
No Grammar:	i saw you film still beat
Learned Grammar:	i saw the movie still beat
Reference Translation:	I saw the movie again
No Grammar:	she loves you the son of her
Learned Grammar:	she likes her son shovel
Reference Translation:	She loves her child
No Grammar:	i cut you solder to bits
Learned Grammar:	i shovels cut the bread to bits
Reference Translation:	I cut the bread into small pieces
No Grammar:	under pressure the bws his it signed with the contract the new
Learned Grammar:	force behind the bws his it signed with the new treaty
Reference Translation:	Under pressure from his boss he signed the new contract
No Grammar:	the drug improved slowly you health his
Learned Grammar:	the drug improved slowly you his health
Reference Translation:	The drug slowly improved his health
No Grammar:	they cuts you the tree the fine
Learned Grammar:	they cuts you the beautiful wood
Reference Translation:	They cut the pretty tree
No Grammar:	i put you the border in case of me
Learned Grammar:	i put you the border with my hand
Reference Translation:	I put the book in my bag

No Grammar: the president be ashamed call you the tablet
Learned Grammar: the president be ashamed shovels read the book
Reference Translation: President Bush read the book

No Grammar: it will wear jackets black
Learned Grammar: he wore black jacket
Reference Translation: He wore a black coat

No Grammar: troops religious h$mi&w you voices their
Learned Grammar: religious groups h$mi&w you their opinions
Reference Translation: Religious groups voiced their opinions

No Grammar: i ate you the swollen yesterday
Learned Grammar: i ate the apple yesterday
Reference Translation: I ate the apple yesterday

No Grammar: we protect activities positive
Learned Grammar: we ensure safe operations
Reference Translation: We guarantee secure transactions

No Grammar: the change the strong in law iiw$m more years
Learned Grammar: the successful transformation of the law iiw$m more years
Reference Translation: The big change in the law will be introduced in two years

No Grammar: troops religious h$mi
w you qwlwthm
Learned Grammar: religious groups h$mi
w you qwlwthm
Reference Translation: Religious groups voiced their opinions

No Grammar: we coat for communication public today
Learned Grammar: we coat for public notice today
Reference Translation: We expect a public announcement today

No Grammar: they started attacks successful
Learned Grammar: they began serious offense
Reference Translation: They started a big attack

No Grammar: you always will be man strong

Learned Grammar:	you always will be strong man
Reference Translation:	You will always be a strong man
No Grammar:	you bestial in many countries
Learned Grammar:	you my animal in many countries
Reference Translation:	I lived in many countries
No Grammar:	it with the way law emphasizes you the importance of patience
Learned Grammar:	it with all access stressed the importance of patience
Reference Translation:	He often emphasizes the importance of patience
No Grammar:	the film show you the fact the very of the adults in the country the it
Learned Grammar:	the film show you the very existence of the adults in the country the it
Reference Translation:	The movie shows the horrible situation of the people in this country
No Grammar:	the police on the road all work tonight
Learned Grammar:	the police of all road work tonight
Reference Translation:	The policeman often works at night
No Grammar:	the soldier write many messages to the family of he
Learned Grammar:	the soldier write many messages to his family
Reference Translation:	The soldier writes many letters to his family
No Grammar:	the president returns with thought his
Learned Grammar:	the president returns for his wisdom
Reference Translation:	The President repeats his opinion
No Grammar:	the mother miss for child of her
Learned Grammar:	the mother miss for her son
Reference Translation:	The mother misses her child
No Grammar:	the doctor helps to patients take out
Learned Grammar:	the assistant doctor for his patients
Reference Translation:	The doctor helps his patients
No Grammar:	this the border most twb

Learned Grammar: it the most border twb
Reference Translation: This is the best book

No Grammar: it built you the tower most important in the country
Learned Grammar: it will build the tower most important in the country
Reference Translation: He built the tallest tower in the country

No Grammar: he students more clerk of the clear
Learned Grammar: it wise more students in the plain
Reference Translation: He is clearly the smarter student

No Grammar: the nominee the second to win support more xzkh of the persons
Learned Grammar: the second candidate to win support more xzkh of the persons
Reference Translation: The second candidate received stronger support from the people

No Grammar: he suggested to him contract longer
Learned Grammar: he suggested if longer treaty
Reference Translation: He offered him a longer contract

No Grammar: the president promised support faster to assemble
Learned Grammar: the president promised stronger support to assemble
Reference Translation: The President promised stronger support for the army

No Grammar: it like ibl scores more successful in exam
Learned Grammar: it like ibl good more on exam results
Reference Translation: He got better results on the test

No Grammar: some for me now achieved clearer regarding the plan of by2
Learned Grammar: some for me now achieved more positive about the plan of by2
Reference Translation: I now have a clearer idea of your plan

No Grammar: the student the navy sweet win awards
Learned Grammar: the student takes the fleet nice cuts
Reference Translation: The nicest student received an award

No Grammar: he appreciated even you the issues most strongly
Learned Grammar: he appreciated even you the most problems difficult

Reference Translation:	He understood even the hardest problems
No Grammar:	he gave to her you the medicine most xzkh
Learned Grammar:	he gave to her you the most drug xzkh
Reference Translation:	He gave her the strongest drug
No Grammar:	weather recognized to be difficult for her
Learned Grammar:	accept weather was difficult for her
Reference Translation:	the cold weather was difficult for her
No Grammar:	the response can to make her military major greatly
Learned Grammar:	the response can to make her military very serious
Reference Translation:	The response could be a very serious military attack
No Grammar:	the book the old contains stories of &ninim greatly
Learned Grammar:	the old book contains stories of &ninim greatly
Reference Translation:	The old book contained very interesting stories
No Grammar:	the film show you the fact the tough greatly for the persons in the country the it
Learned Grammar:	the film show you the very harsh reality of the adults in the country the it
Reference Translation:	The movie shows the extremely difficult situation of the people in this country
No Grammar:	the book the back of the hebrew contains many matters of &ninim and humorous
Learned Grammar:	the old book on hebrew contains many matters of &ninim and humorous
Reference Translation:	The old Hebrew book contained many interesting stories
No Grammar:	i put you the border in case of me the fast
Learned Grammar:	i put you the barber in my case the fast
Reference Translation:	I quickly put the book in my bag
No Grammar:	students youths on the road all want to learn countries learn and for people learn
Learned Grammar:	young students with access all wish to study various countries and with different peoples

Reference Translation:	Young students often want to learn about different countries and about different people.
No Grammar:	it like ibl products successful in exam
Learned Grammar:	it like ibl good results of examination
Reference Translation:	He got good results on the test
No Grammar:	any called you it on the borders and in newspaper
Learned Grammar:	any called this shovel of borders and in newspaper
Reference Translation:	i read this in books and in the newspaper
No Grammar:	the wife of dress the human lives of many countries
Learned Grammar:	the wife of the red dress lives of many countries
Reference Translation:	The woman in the red dress lived in many countries
No Grammar:	i slept when he read you the tablet
Learned Grammar:	i slept when he read the book
Reference Translation:	I slept while he read the book

Bibliography

Aho, A. V., & Ullman, J. D. (1969). Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, *3*, 37–56.

Alberto Alonso, J. (1990). Transfer InterStructure: designing an 'interlingua' for transfer-based MT systems. *Proceedings of the 3rd International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language*.

Allen, J. (1995). *Natural Language Understanding, 2nd Edition*.

Alshawi, H., Bangalore, S., & Douglas, S. (1998). Automatic Acquisition of Hierarchical Transduction Models for Machine Translation. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL-COLING-1998)*.

Alshawi, H., Douglas, S., & Bangalore, S. (2000). Learning Dependency Translation Models as Collections of Finite-State Head Transducers. *Computational Linguistics*, *26*.

Ayan, F., Dorr, B. J., & Habash, N. (2004). Application of Alignment to Real-World Data: Combining Linguistic and Statistical Techniques for Adaptable MT. *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA-04)*.

Boitet, C. (1988). Bernard Vauqois' contribution to the theory and practice of building MT systems: a historical perspective. *Proceedings of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*.

Bouquiaux, L., & Thomas, J. (1992). *Studying and Describing Unwritten Languages*. Dallas, TX: The Summer Institute of Linguistics.

Bresnan, J. (2001). *Lexical-Functional Syntax*. Blackwell Textbooks in Linguistics.

Brown, P., Della Pietra, V. J., Della Pietra, S. A., & Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, *19*, 263–311.

Brown, R. (1997). Automated Dictionary Extraction for 'Knowledge-Free' Example-Based Translation. *Proceedings of the 7th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-97)*.

Charniak, E. (2000). A Maximum-Entropy-Inspired Parser. *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics and 6th Conference on Applied Natural Language Processing (NAACL/ALP-00)*.

Charniak, E., Knight, K., & Yamada, K. (2003). Syntax-based Language Models for Statistical Machine Translation. *Proceedings of the 9th Machine Translation Summit (MT-Summit IX)*.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*.

Collins, M. (1996). Identifying head-words in the WSJ Penn treebank.

Collins, M. (2003). Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*, *29*.

Comrie, B., & Smith, N. (1977). Lingua Descriptive Series: Questionnaire. *Lingua*, *42*, 1–72.

Dahan, H. (1997). Hebrew-English English-Hebrew Dictionary.

Dorr, B. J. (1992). *Machine Translation: A View from the Lexicon*. MIT Press.

Dorr, B. J., Ayan, N. F., & Habash, N. (2004). Divergence Unraveling for Word Alignment of Parallel Corpora. *submitted to Journal of Natural Language Engineering*.

Dorr, B. J., Ayan, N. F., Habash, N., & Hwa, R. (2003). Rapid Porting of DUSTer to Hindi. *ACM Transactions on Asian Language Information Processing (TALIP)*, *2*.

Dorr, B. J., Pearl, L., Hwa, R., & Habash, N. (2002). DUSTer: A Method for Unraveling Cross-Language Divergences for Statistical Word-Level Alignment. *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas (AMTA-02)*.

Falk, Y. N. (2004). The Hebrew Present-Tense Copula as a Mixed Category. *Proceedings of the Lexical Functional Grammar 04 Conference (LFG-04)*.

Font-Llitjós, A. (2004). Towards Interactive and Automatic Refinement of Translation Rules. *Thesis Proposal*.

Font-Llitjós, A., Probst, K., & Carbonell, J. (2004). Error Analysis of Two Types of Grammar for the Purpose of Automatic Rule Refinement. *Proceedings of the 6th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-04)*.

Goldsmith, J. (2001). Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, *27*, 153–198.

Guvenir, H. A., & Tunç, A. (1996). Corpus-Based Learning of Generalized Parse Tree Rules for Translation. In G. McCalla (Ed.), *New Directions in Artificial Intelligence: Proceedings of the 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*. Toronto, Canada: Springer Verlag.

Habash, N. (2002). Generation-Heavy Hybrid Machine Translation. *Proceedings of the 2nd International Natural Language Generation Conference (INLG-02)*.

Habash, N., & Dorr, B. J. (2002). Handling Translation Divergences: Combining Statistical and Symbolic Techniques in Generation-heavy Machine Translation. *Proceedings of the 5th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-02)*.

Habash, N., & Dorr, B. J. (2003). CatVar: A Database of Categorial Variations for English. *Proceedings of the 9th Machine Translation Summit (MT-Summit IX)*.

Hermjakob, U., & Mooney, R. (1997). Learning Parse and Translation Decisions From Examples With Rich Context. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL/EACL-97)*.

http://www.computing.dcu.ie/~acahill/tagset.html (1990). Penn Treebank Tagset.

Hutchins, W. J., & Somers, H. L. (1992). *An Introduction to Machine Translation*. London: Academic Press.

Hwa, R. (1999). Supervised Grammar Induction using Training Data with Limited Constituent Information. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*.

Itai, A., & Segal, E. (2003). A corpus based morphological analyzer for unvocalized modern hebrew. *Proceedings of Machine Translation for Semitic Languages: Issues and Approaches, Workshop at MT Summit IX (MT-SUMMIT-IX)*.

Johnson, H., & Martin, J. (2003). Unsupervised Learning of Morphology for English and Inuktitut. *Proceedings of the Human Language Technology Conference / Conference of the North American Chapter of the Association for Computation Linguistics 2003 (HLT/NAACL-03)*.

Jones, D., & Havrilla, R. (1998). Twisted Pair Grammar: Support for Rapid Development of Machine Translation for Low Density Languages. *Proceedings of the 3rd Biennial Conference of the Association for Machine Translation in the Americas (AMTA-98)*.

Kaji, H., Kida, Y., & Morimoto, Y. (1992). Learning Translation Templates from Bilingual Text. *Proceedings of the 15th International Conference On Computational Linguistics (COLING-92)*.

Klein, D., & Manning, C. (2001). Distributional Phrase Structure Induction. *Proceedings of the Fifth Conference on Natural Language Learning (CoNLL-2001)*.

Knight, K., Al-Onaizan, Y., Chander, I., Hovy, E., Langkilde, I., Whitney, R., & Yamada, K. (1996). System demonstration: JAPANGLOSS: using statistics to fill knowledge gaps. *Expanding MT horizons: Proceedings of the Second Conference of the Association for Machine Translation in the Americas (AMTA-96)*.

Knight, K., Chander, I., Haines, M., Hatzivassiloglou, V., Hovy, E., Iida, M., Luk, S. K., Whitney, R., & Yamada, K. (1995). Filling knowledge gaps in a broad-coverage MT system. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-1995)*.

Lavie, A., Sagae, K., & Jayaraman, S. (2004). The Significance of Recall in Automatic Metrics for MT Evaluation. *Proceedings of the 6th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-04).*

Lavie, A., Vogel, S., Levin, L., Peterson, E., Probst, K., Font-Llitjós, A., Reynolds, R., Carbonell, J., & Cohen, R. (2003). Experiments with a Hindi-to-English Transfer-based MT System under a Miserly Data Scenario. *ACM Transactions on Asian Language Information Processing (TALIP), 2.*

Lavoie, B., White, M., & Korelsky, T. (2002). Learning Domain-Specific Transfer Rules: An Experiment with Korean to English Translation. *Proceedings of the Workshop on Machine translation in Asia, 19th International Conference on Computational Linguistics.*

Leech, G. (1992). 100 Million Words of English: the British National Corpus. *Language Research, 28.*

Leech, G., Garside, R., & Bryant, M. (1994). CLAWS4: The tagging of the British National Corpus. *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94).*

Levin, L., & Nirenburg, S. (1994). Construction-Based MT Lexicons. *Current Issues in Computational Linguistics: In Honour of Don Walker. Zampolli, Calzolari, and Palmer (eds.).*

Marcus, M., Taylor, A., MacIntyre, R., Bies, A., Cooper, C., Ferguson, M., & Littmann, A. (1995). *The Penn Treebank Project.* http://www.cis.upenn.edu/~treebank/home.html.

McCormick, S. (1998). A centralized approach to managing multiple lexical resources. *Proceedings of the 3rd Workshop of the Euroapean Association for Machine Translation (EAMT-98).*

Menezes, A., & Richardson, S. D. (2001). A Best-first Alignment Algorithm for Automatic Extraction of Transfer Mappings from Bilingual Corpora. *Proceedings of the Workshop on Data-driven Machine Translation at the 39th Annual Meeting of the Association for Computational Linguistics, (ACL-01).*

Meyers, A., Yangarber, R., Grishman, R., Macleod, C., & Moreno-Sandoval, A. (1998). Deriving Transfer Rules from Dominance-Preserving Alignments. *Proceedings of the 36th Annual Meeting of the Association for*

Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL-98).

Monson, C., Lavie, A., Carbonell, J., & Levin, L. (2004). Unsupervised Induction of Natural Language Morphology Inflection Classes. *Proceedings of the Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON) at the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04).*

Needleman, S., & Wunsch, C. (1970). A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology*, *48*, 443–453.

Nirenburg, S. (1998). Project Boas: A Linguist in the Box as a Multi-Purpose Language Resource. *Proceedings of the First International Conference on Language Resources and Evaluation (LREC-98).*

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., & Radev, D. (2003). Syntax for Statistical Machine Translation.

Och, F. J., & Ney, H. (2002). Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL-02).*

Och, F. J., & Ney, H. (2004). The alignment template approach to statistical machine translation. *Computational Linguistics*, *30*.

Papineni, K., Roukos, S., & Ward, R. T. (1998). Maximum Likelihood and Discriminative Training of Direct Translation Models. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP-98)* (pp. 189–192).

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2001). *Bleu: A Method for Automatic Evaluation of Machine Translation* (Technical Report RC22176(W0109-022)). IBM Watson Research Center.

Peterson, E. (2002). Adapting a Transfer Engine for Rapid Machine Translation Development. Master's thesis, Georgetown University.

Probst, K. (2002). Semi-Automatic Learning of Transfer Rules for Machine Translation of Low-Density Languages. *Proceedings of the Student Session*

at the 14th European Summer School in Logic, Language and Information (ESSLLI-02).

Probst, K. (2003). Using 'Smart' Bilingual Projection to Feature-tag a Monolingual Dictionary. *Proceedings of the 7th Conference on Natural Language Learning (CoNLL-03).*

Probst, K., Brown, R., Carbonell, J., Lavie, A., Levin, L., & Peterson, E. (2001). Design and Implementation of Controlled Elicitation for Machine Translation of Low-density Languages. *Proceedings of the MT2010 Workshop at the Machine Translation Summit VIII (MT-Summit VIII).*

Probst, K., & Lavie, A. (2004). A Structurally Diverse Minimal Corpus for Eliciting Structural Mappings between Languages. *Proceedings of the 6th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-04).*

Probst, K., & Levin, L. (2002). Challenges in Automated Elicitation of a Controlled Bilingual Corpus. *9th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-02).*

Probst, K., Levin, L., Peterson, E., Lavie, A., & Carbonell, J. (2003). MT for Minority Languages Using Elicitation-Based Learning of Syntactic Transfer Rules. *Machine Translation, Special Issue on Embedded MT.*

Sato, S., & Nagao, M. (1990). Towards Memory-based Translation. *Proceedings of the 13th International Conference On Computational Linguistics (COLING-90).*

Schone, P., & Jurafsky, D. (2001). Knowledge-Free Induction of Inflectional Morphologies. *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-01).*

Segal, E. (2001). A Hebrew morphological analyzer (includes free source code).

Senellart, J., Plitt, M., Bailly, C., & Cardoso, F. (2001). Resource Alignment and Implicit Transfer. *Proceedings of the Machine Translation Summit VIII (MT-Summit VIII).*

Sherematyeva, S., & Nirenburg, S. (2000). Towards a Universal Tool for NLP Resource Acquisition. *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-00).*

Shieber, S. M. (1986). An Introduction to Unification-Based Aproaches to Grammar. In *Lecture notes*, vol. 4. Center for the Study of Language and Information.

Snover, M. G., & Brent, M. R. (2002). A Probabilistic Model for Learning Concatenative Morphology. *Proceedings of the 16th Annual Conference on Neural Information Processing Systems (NIPS-02).*

Steiner, E. (1990). Aspects of a functional grammar for machine translation. *Proceedings of the 3rd International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language (TMI-90).*

Thurmair, G. (1990). Complex lexical transfer in METAL. *Proceedings of the 3rd International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language (TMI-90).*

Vogel, S., Zhang, Y., Huang, F., Tribble, A., Venogupal, A., Zhao, B., & Waibel, A. (2003). The CMU Statistical Translation System. *Proceedings of the Machine Translation Summit IX (MT-Summit IX).*

Watanabe, H., Kurohashi, S., & Aramaki, E. (2000). Finding Structural Correspondences from Bilingual Parsed Corpus for Corpus-based Translation. *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00).*

Wu, D. (1997). Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, *23*.

Xia, F., & McCord, M. (2004). Improving a Statistical MT System with Automatically Learned Rewrite Patterns. *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04).*

Yamada, K., & Knight, K. (2001). A Syntax-Based Statistical Translation Model. *Proceedings of the 39th Anniversary Meeting of the Association for Computational Linguistics (ACL-01).*

Yamada, K., & Knight, K. (2002). A Decoder for Syntax-Based Statistical MT. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL-02).*

Yarowsky, D., & Ngai, G. (2001). Inducing Multilingual POS Taggers and NP Bracketers via Robust Projection Across Aligned Corpora. *Proceedings*

of the Second Meeting of the North American Chapter or the Association for Computational Linguistics (NAACL-2001) (pp. 200–207).

Yarowsky, D., Ngai, G., & Wicentowski, R. (2001). Inducing Multilingual Text Analysis Tools via Robust Projection across Aligned Corpora. *Proceedings of the First International Conference on Human Language Technology Research (HLT-01).*

Yarowsky, D., & Wicentowski, R. (2000). *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-00).*

Zhang, H., & Gildea, D. (2004). Syntax-Based Alignment: Supervised or Unsupervised? *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04).*

Zhang, Y., & Vogel, S. (2004). Measuring Confidence Intervals for the Machine Translation Evaluation Metrics. *Proceedings of The 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-04).*