ADAPTING A TRANSFER ENGINE FOR RAPID DEVELOPMENT MACHINE TRANSLATION

Master's Research Paper
Department of Linguistics
Georgetown University

By

Erik Edwin Peterson, B.S.

Washington, DC
May 1, 2002

Contents

Chapter 1

Introduction

1.1 Overview

Early machine translation (MT) systems required manual development of linguistic databases and large numbers of rules to describe the translation process. The necessary outlay of time and effort generally limited such systems to commonly-spoken or economically significant language pairs, neglecting systems for less commonly spoken languages that could benefit from automatic translation. However, if rules for such systems could be automatically learned from minimal amounts of bilingual corpora, MT systems for these otherwise ignored language pairs could be developed. Such is the goal of

the AVENUE project. This paper describes the background, rule formalism, and algorithms for a translation engine written especially for the AVENUE project and details the adaptations necessitated by the training process. The engine is designed to assist this training process, as well as to use the learned rules for a final machine translation system. The translation engine itself uses a variation of the transfer approach to MT and a background on previous transfer systems is included.

The type of data-driven machine translation in AVENUE differs from Example-based MT (EBMT) and Statistical MT in that training produces a set of transfer rules that can then be used to translate like a standard transfer engine. The automatically learned rules can later be manually altered or added to as needed. The training process itself will be described briefly.

1.2 AVENUE Project

The AVENUE project seeks to combine the advantages of data-driven MT and manually-developed transfer systems. Data-driven approaches, such as EBMT or Statistical Machine Translation, use large amounts of aligned parallel text in the source and target languages to train models which are then

used in the actual translation process. After development of the training software, comparatively little human effort is needed to train a new model and run translations. In contrast, transfer-based systems require time-consuming manual development of the system's linguistic data and processing rules, both syntactic and lexical. Example text is used as a guide by the rule developers, but large amounts of training corpora are not necessary.

The AVENUE approach (called Instructible Rule-Based MT) is detailed in Probst, et.al. (2001):

> For iRBMT a bilingual user who is not an expert in linguistics is asked to translate a set of sentences and specify the word alignment between source and target language sentences. The goal of the learning process is to match every translation example in the elicited bilingual corpus with a transfer rule that accounts for the translation and is of an appropriate level of abstraction. For instance, after the system observes the translations of several example noun phrases of similar structure but containing different nouns and adjectives, it will automatically infer that the transfer rules of the examples can be collapsed into a single transfer rule.

The set of English sentences to be translated, developed by Kathrina

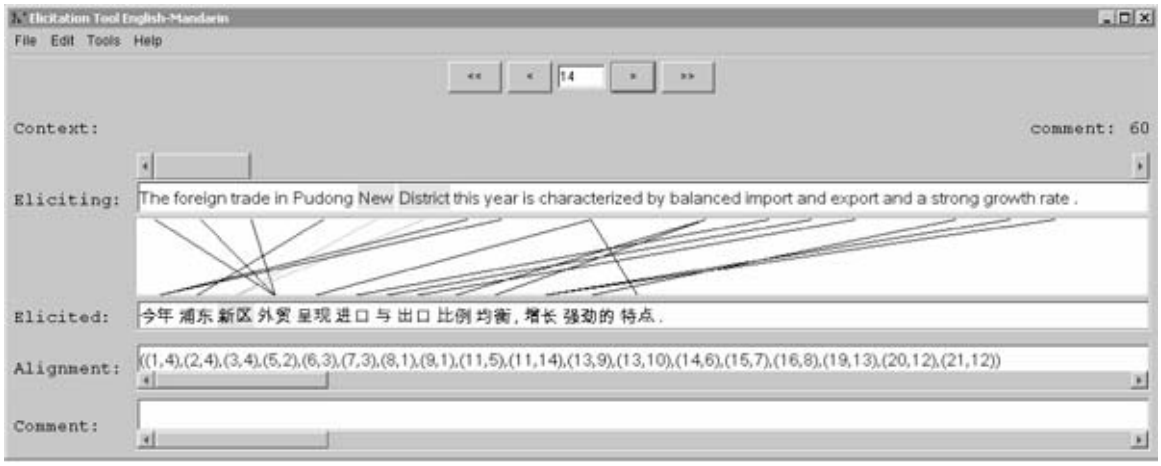


Figure 1.1: Elicitation Interface

Probst and called the elicitation corpus, has been carefully selected to include a wide variety of possible linguistic phenomena such as "word order (within noun phrases and clauses), grammatical features (number, gender, person, tense, definiteness), and agreement patterns" (Probst et.al., 2001). The corpus currently includes about 800 sentences. Each elicitation sentence is pre-parsed and its analysis disambiguated. Pre-analysis requires a parser for the elicitation sentences, an example being LCFlex for English (Rose and Lavie, 2001). The elicitation corpus has also been translated into Spanish. The AVENUE project is currently planning to work with the target languages Mapudungun in Chile and Inupiak in Alaska.

The sentences are presented to the informant through an specially-designed elicitation interface (Figure 1.1). The informant translates each sentence and provides the word-level alignments between the eliciting language sentence and the translated sentence. Minimal pairs of sentences differing in just one attribute (such as one noun being plural and another singular) allow the elicitation tool to detect the linguistic features marked in the target language. The translations and alignments can influence the future sentences shown to the informant. For example, if the language does not distinguish singular and plural nouns, then the system does not need to check for a dual form. This feature detection can also help determine what features to use in the learned transfer rules.

The word-aligned sentences and annotated source language sentences are then used to produce highly-specific "seed" transfer rules that can translate the input sentence pair. The format of the rules is explained in chapter 3, but in short they include both syntactic and feature transfer information.

The seeds are input into a variant of the version space learning algorithm (Mitchell, 1982) which attempts to combine rules to produce generalized rules that will not only translate the elicitation sentences, but also previously unseen data. In this way, the coverage of the final system is maximized. Each

seed rule is associated with a set of sentences that it covers (i.e. translates correctly) called its CSet (Probst, unpublished). Two rules are candidates for merging if they translate the same source constituent types to the same target constituent types in the same order. The rules are combined by deleting or generalizing feature constraints and then testing to see if the combined rule's CSet includes both seed rules' CSets by running the rule through the transfer engine and checking the correctness of the output. This check can be done either through a simple string match or by having a native informant judge the output. Rule generalization is repeated until a set of rules is produced that is as general as possible but still produces correct translations of the input sentence pairs. These rules form the basis of the final system.

Chapter 2

Transfer Overview

2.1 Background on MT Approaches

Most traditional approaches to machine translation vary in the amount of analysis done on the source language side and in the abstractness of the intermediary representation. Both affect the effort needed to produce a target language representation and generate the final translation. Some systems are hybrids of various approaches. The relatively recent approaches of example-based and statistical machine translation are not directly relevant for this paper and will not be discussed here. The well-known Vauquois Triangle (Figure 2.1) summarizes this relationship between analysis, intermediate translation

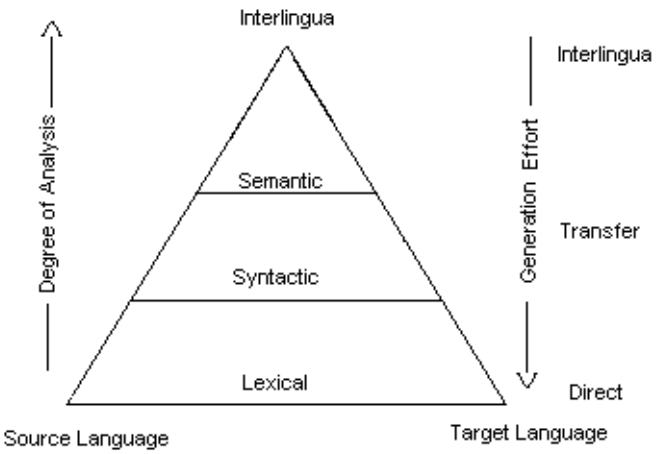


Figure 2.1: Vauquois Triangle

effort, and target language generation.

At the bottom of the triangle is the simplest form of MT and the type used by many early translation systems: the direct approach. In the direct approach only limited morphological and lexical analysis occur on the source language sentence. No deeper syntactic or semantic structures are created and the limited analysis might even be affected by the target language. The words are then directly translated to their target forms, with limited lexical reordering to better conform to the target language syntax. No intermediate

translation representation is used. The Georgetown University GAT system and the early Systran system that grew out of GAT (Hutchins, 1992) are both examples of the direct approach.

Direct MT systems are relatively easy to write and require little source language analysis. However, this shallow analysis and simple generation reduces the systems' ability to translate syntactic structures that differ between languages and handle various ambiguities that might only be resolved by a deeper analysis. Also, since the source text is translated directly into the target language, very little of the system for a language pair can be reused when a translation system is to be written using one of the languages in the pair with a different language.

In contrast, the high end of the triangle requires extensive analysis of the source language text into a language-neutral form, called an interlingua. Conceptually, an interlingua representation can be used to translate to any other language that has a generator from the interlingua to a surface form. Systems such as the KANT project at Carnegie Mellon University (Nirenburg, 1992) and Distributed Language Translation from the software company Buro voor Systemontwikkeling (BSO) (Hutchins, 1992), are examples of the interlingua approach to machine translation.

While interlinguas can make writing multilingual systems much simpler by lowering the bar for adding new language pairs to a translation system, significant difficulties exist for developing general coverage interlingua MT systems. Analysis to the level of the interlingua requires many resources to resolve ambiguities inherent in all language. Additionally, defining an interlingua powerful enough to represent the breadth of possible sentence meanings across various languages is problematic. Some approaches have been to use an intermediate human language, such as Esperanto in the DLT system, or to develop a logical knowledge representation scheme as used in KANT. Interlingua systems have been successful in translating text drawn from a limited domain, such as heavy machinery repair manuals, but do not yet work well in unrestricted contexts.

2.2 Transfer-based MT

The inaccuracies of direct MT and the development effort and resources needed for interlingua MT have led most commercial MT systems to use an intermediate approach: transfer. In the transfer approach to MT the source text is analyzed into an abstract representation that still has many of the

characteristics of the source, but not the target, language. This representation can range from purely syntactic to highly semantic.

In syntactic transfer, the parse tree of the source input is altered by some type of tree manipulation into a target language tree. This can be guided by associating feature structures with the tree. A feature structure (or f-structure) is a set of feature (also called attribute) value pairs, where a value can itself be an f-structure. F-structures store morphological, syntactic and semantic data for the source sentence. Some systems forgo using the syntactic structure and transfer directly on the f-structure. Other systems attempt a more in depth analysis to produce a semantic representation of the source sentence, perhaps using first-order predicate logic as the representation.

Whatever representation is used, transfer to the target language is done using rules that map the source language structures into their target language equivalents. Then in the generation stage, the mapped target structure is altered as required by the constraints of the target language and the final translation is produced. Traditionally, transfer systems have taken years to produce because they need hundreds of manually crafted rules and thousands of lexical entries in order to deal with even a small subset of the potential input.

2.3 Existing Transfer Systems

2.3.1 TAUM-MÉTÉO

One of the earliest and most successful transfer-based systems is the MÉTÉO system developed by the TAUM group in Montreal. MÉTÉO translates weather bulletins from English into French, a very limited domain that allows for the system's high accuracy. The TAUM system is notable for strictly separating the transfer algorithms and the linguistic data (Hutchins, 1986). Language data is not hard-coded into the actual transfer software, leading to more flexibility and easier maintenance. As also detailed by Hutchins, the system is a "typical example of the 'transfer' approach in perhaps the purest form, having five basic stages: Morphological analysis of English, Syntactic analysis of English, Transfer, Syntactic generation of French, Morphological generation of French." However, the TAUM system operates only on the sentence level since it lacks any means for discourse analysis. It also has no back-up strategy for sentences that do not fully parse or transfer; nothing at all is output.

In MÉTÉO's morphological analysis words are assigned category labels (e.g. part of speech tags), analyzed into stems and affixes, and assigned

features (such as animate, abstract, etc.). Syntactic analysis then identifies noun phrases and complex verb forms. These phrase structure and transformation rules are not only able to identify source language syntactic structures, but also manipulate them, perhaps changing a passive sentences to its active form. Some semantic information is used to choose possible alternate structures.

Transfer starts with lexical transfer, where a bilingual dictionary is used to translate source base word forms to target language equivalents, which could also involve manipulating the tree itself. The system also does structural transfer, altering the tree. Generation involves breaking the complex post-transfer tree into smaller trees which are then converted through morphological generation to their surface forms. Hutchins concludes "the TAUM system illustrates well the characteristic features of MT transfer systems: the clear separation of the different stages of analysis and synthesis, the separation of linguistic data from processing algorithms..., and the use of separate dictionaries for analysis, transfer, and generation."

2.3.2 METAL

The METAL system (short for 'Mechanical Translation and Analysis of Language') as described in Hutchins (1992), was originally developed at the Linguistics Research Center at the University of Texas at Austin during the 1960's and 70's and later incorporated into Siemens (and then Lernout and Hauspie). It started with a German to English system, later branching to Dutch, French, and several other European languages. METAL's goal was to produce translated text that would be of decent quality but still possibly need to post-editing. METAL is also notable for its lexical and transfer rule development tools and its ability to handle formatted text that included diagrams, tables, charts, etc.

METAL has three tiers of dictionaries, one for function words, another for general vocabulary, and a third that can be tuned to whatever technical vocabulary is needed for a given text. Source and target dictionary entries include "lists of features with values, e.g. root form, grammatical category, morphological variants, number, person, etc." (Hutchins, 1992) Multiple entries for a word form are ranked by 'preference'. The source and target dictionaries are designed to be monolingual and uninfluenced by the other language in the pair. The bilingual dictionary is specific to a particular

language pair and includes the source word, feature values to be matched, and a target word to use in the case of a match. A source word can have more than one potential translation. Lexical ambiguity is resolved by feature value matches and by an explicit ordering of possible translations, with a default at the end.

Grammars in the first version of METAL were "unordered sets of context free phrase structure rules augmented by tests and conditions and by specifications of the structures to be output." (Hutchins, 1992) These structures are both syntactic and functional. Analysis grammars may also include rules to rearrange or transform source language trees. Each analysis rule also includes transfer conditions to be checked and mappings to be applied during transfer at the tree node created by the rule during analysis. This close integration of the analysis, transfer, and generation rules led Hutchins to call the early METAL system a 'modified transfer system'. Work has since been done to decouple the analysis and transfer rules to produce a 'pure' transfer system.

Translation starts with a bottom-up chart parse of the input text. Grammar rules are prioritized with preferential weightings. The chart allows for partial parses to be retained and used later for transfer if a complete parse

is not found. Parses generally result in fairly shallow trees. Transfer is done from the top node down using the transfer section of the analysis rule. If all conditions are met, the tree is reordered as directed by the rule. Transfer progresses recursively down the original source tree until a target tree is generated. The final translation can then be read directly from the lexical leaves of the transferred tree, after any final generational morphology.

2.3.3 GETA Ariane

The influential Ariane system, first developed during the 1970's by the GETA group at the University of Grenoble in France headed by Bernard Vauquois, is a 'pure' transfer system with analysis, transfer and generation stages and rules all cleanly separated (Hutchins 1992). Different stages of translation are handled by different software packages with their own formalisms, including a package for morphological analysis and generation, another for lexical transfer, and another tree manipulation package that handles structural analysis, transfer and generation.

Translation starts with a morphological analysis on the source text words producing the stems and a set of associated feature value pairs. This is followed by syntactic analysis to a multi-level linguistic representation with

morphological information at the word level, syntactic information like noun or verb phrase groups, and 'logico-semantic information' showing "dependency relations (heads and modifiers) of a logical nature, e.g. predicates and arguments or circumstantial elements with their semantic roles (goal, cause, location, etc.)" (Hutchins, 1992). This representation is first used for lexical transfer and then structural transfer. Structural transfer is guided by tree mapping rules activated when specified conditions are met. This transferred representation is then modified again by tree mapping rules to final target language structures. In the last stage, the lexical items in the tree are run through generational morphology to produce their surface forms. The final translation is then returned.

2.3.4 Eurotra

Eurotra was a project funded by the then European Economic Community to develop high-quality machine translation between the seven languages of the EEC: initially Danish, Dutch, English, French, German, Greek and Italian, with Spanish and Portuguese added later. This description of the early stages of Eurotra is drawn from Arnold, 1987. Given the decision to use the transfer approach in Eurotra, full implementation required 72

separate transfer modules, along with the nine analysis and nine generation components. Work on this was distributed amongst various groups in the nine countries, helping to develop computational linguistic expertise among the member nations.

In light of the large distributed nature of Eurotra, the participants developed a highly modularized framework and standardized intermediate representation structure to be fed to all the various transfer modules. Analysis used a unification-based grammar formalism and had five intermediate steps: extraction of text from a formatted document, morphological analysis on individual words, construction of a syntactic constituent representation, construction of a feature value structure with grammatical relations between phrases, and finally the "Interface Structure" or IS, which included semantic information on θ-roles and typing (animacy, abstractness, etc.). Given the amount of transfer modules necessary, the developers decided to simplify transfer as much as possible, leading to the need for the higher levels of analysis seen in the IS.

Translation was done in a series of "generative devices" and "translators", unfortunately named steps as they are not actually directly related to generation or translation. A generative device is a pair <C, A> of a "con-

structor" and set of "atoms". Constructors, or rules, "will take a number of expressions as arguments, and return a new expression". Atoms are linguistic constructs, such as syntax tree nodes or lexical items. Constructors can manipulate atoms by deleting elements, rearranging structures, introducing new elements, or modifying feature value groups. The output of a generator is then mapped by the translator T into the representation expected by the next generator. A series of these <C,A>,T groups is used for analysis, transfer, and generation to produce a final translation.

2.3.5 Verbmobil

Verbmobil is a semantic-transfer system primarily used for German, English and Japanese speech translation for face-to-face dialogs. A project of the German Federal Ministry of Education, Science, Research and Technology, Verbmobil development occurred mostly during the 1990's. Written in Prolog, the software "transforms a semantic representation that is the input to generation for a target language. Therefore, the transfer equivalences abstract away from morphological and syntactic analyses of source and target languages" (Dorna, 1996). This semantic representation, based on Underspecified Discourse Representation Structures, can contain ambiguity in

quantifier and operator scope which allows for ambiguity preserving translations. The representation also includes information on tense, aspect, prosody and morpho-syntax. Given the use of the system for speech translation, discourse elements can also be included.

Transfer is done with rules that map between sets of source and target language semantic entities. A rule has five main parts, detailed in Dorna, 1996. SLSem contains the source semantic entities that will be matched and mapped to TLSem, the target language semantic entities. The application of a rule is governed by SLConds, which are semantic conditions that must be matched in the source representation but themselves will not be mapped, and TLConds which are similar conditions on the target side. The intended application direction can also be specified, though it is usually from source to target. Unlike METAL, application does not specifically trigger recursive application of other rules. The sequence of rule application is governed by "specificity ordering", where rules with larger SLSem and SLConds are preferred over those with fewer conditions. After transfer, the target representation is then converted to the target language surface form.

2.3.6 TranSphere

The TranSphere system claims to be one of the first MT systems to be based on Lexical Functional Grammar (Bresnan, 2001). TranSphere originated at Executive Computer Systems (ECS), which created it in the late 1980's. The system was later purchased by Applications Technologies (AppTek) of McLean, Virginia. The system first parses the source text to two levels of linguistic information. The first, constituent or c-structure, is the syntactic representation of the text. The second, functional or f-structure, is "a hierarchical attribute-value matrix representing the relatively language-independent underlying grammatical relations of a sentence, serves ideally as the basis for transfer, while the highly language-dependent constituent structure is discarded" (Her, 1991). The lexicon includes a mix of semantic and world knowledge that can be used to select correct lexical translations given the f-structure. The f-structure is then mapped to a target f-structure and generation to the target text is completed from the mapped f-structure.

2.4 Recent Training-related Transfer Systems

Like AVENUE, the systems described below rely on there being a grammar for one of the languages in a pair, and somehow learning correspondences between the language structures. These correspondences are then used during translation.

2.4.1 CRL Expedition

The goal of the Expedition project at the Computing Research Laboratory (CRL) of New Mexico State University is to produce a "semi-automatic knowledge elicitation system that guides a team of two people [a programmer and language specialist] through the process of developing the static knowledge sources for a moderate-quality, broad-coverage MT system from any 'low-density' language in about six-months" (Nirenburg, 1998). The linguistic elicitation was part of the Boas sub-project while the machine translation system used in Expedition came from the CRL Corelli project. Since the CRL system was designed to work with minimal resources and to translate from as many sources as possible, it emphasizes coverage and robustness over depth of analysis (Zajac, 1999), allowing for the systems to produce results almost from the beginning of development and to improve quality as

more information is added. The Corelli system also emphasized the use of a modularized design with well-defined interfaces among modules.

The system attempts robustness through a large lexicon, robust subcomponents (such as a morphological analyzer that can handle misspellings and proper names), and various levels of back-off, so that if one components fails to produce results, the output of the previous component can be used (possibly degrading all the way to direct word to word translation).

The Corelli system starts with a morphological analysis of the words, producing all possible segmentations of each word to be used in later processing. This eliminates any problems that would arise later from choosing just one (possibly incorrect) analysis. Analysis identifies lexical compounds, modifiers and specifiers, complements, clauses and parallel structures, possibly recursively. Grammar rules consist of a right-hand side with a pattern to be matched, a left-hand side describing the structure to be built (including how features should be passed from the right to left-hand side structures), and Boolean constraints on acceptable feature structures. Whenever a high-level constituent is identified, the subconstituents composing it are deleted.

Transfer is divided into three parts: lexical transfer for words that have not been incorporated into a syntactic constituent, limited non-recursive lex-

ical reordering to target surface order (also for words not in a syntactic structure), and a possibly recursive structural transfer component. The structural transfer will "produce arbitrary target mopho-syntactic structures, introducing for example syntactic phenomena that are morphological phenomena in the source language" (Zajac, 1999). Structural transfer and reordering "are interleaved so that structural transfer can be called on a sub-constituent produced by a reordering rule" (Zajac, 1999). System development can start with lexical information and add structural rules incrementally.

2.4.2 Stochastic Grammar Channel (SGC)

While not a transfer-based MT system, the Stochastic Grammar Channel (also called Stochastic Inverse Transduction Grammar or SITG) of Dekai Wu (Wu, 1998) has several interesting similarities to trained rule-based systems. The system currently translates from English to Chinese. SGC is a combination of statistical MT and grammatical MT. In statistical MT, a model is built of the translation to the target language including target word selection and alignment as well as a language model of the target language. Normally, the target language model uses only word bigrams to generate output, often leading to ungrammatical translations.

Wu's Grammatical Channel Model improves on the word alignment model by using a grammatical association between source and target languages. The assumption is that a modified "mirrored version of the target language can parse sentences of the source language" (Wu, 1998). The modifications include allowing reversal on the order of the constituents on the right-hand side of production rules, mapping parts of speech between the two languages, and allowing for word skipping. This Inverse Transduction Grammar (ITG) is used to parse the source language with the result that "any resulting parse tree must be consistent with the original Chinese grammar." Translation occurs by parsing the source language with the ITG from the target language which simultaneously generates a target language parse tree. The best target parse tree is determined with a target language model using bigrams and production rule probabilities. In a sense, the ITG allows for grammatical and lexical transfer between the source and target language.

2.4.3 ALT-J/E

The ALT-J/E system's goal is to translate from Japanese to English. While primarily built with manually constructed rules, the developers found that such manual rule building was costly in time and effort, and incorporating

new rules could interfere with output from existing rules. In Akiba et.al., 1995, the authors describe their efforts to automatically learn English verb selection rules from hand-crafted translation rules and a small number of translation examples. The hand-crafted rules contain semantic constraints on the arguments to the Japanese pair and the corresponding English translation given those constraints. Using the translation examples, the system is able refine these rules to increase their accuracy in the context of the existing set of rules. No mention is made in the paper of the translation approach used by the system, though it seems to be transfer.

Chapter 3

AVENUE Transfer Algorithm

3.1 Transfer Rules Formalism

Most of the transfer engines described in the previous chapter have separate rules for the three traditional stages of transfer MT: analysis, transfer, and generation. Due to the training constraints, the AVENUE transfer engine combines the information for all three stages into a single rule, but uses it in several stages. The AVENUE engine would consequently be closer to the 'modified' transfer approach used in the early METAL system as described by Hutchins (1992). Like all of the systems previously mentioned, the AVENUE transfer engine maintains a strict separation between the transfer algorithms

and linguistic data. The design of the transfer rule formalism itself was guided by the consideration that the rules must be simple enough to be learned by an automatic process, but also powerful enough to allow manually-crafted rule additions and changes to improve the automatically learned rules. This means that the knowledge-intensive semantic transfer approach used in Verbmobil would be impractical for AVENUE. Transfer on the feature structure alone, as in TranSphere, would also be difficult.

The input to the training system is a set of word-aligned bilingual sentences, with a syntactic parse and feature structure available only for the source sentences. The source sentence parse and word alignments are used learn syntactic rules. Using the source f-structure and target features (learned during elicitation), these syntactic transfer rules can be guided by feature constraints. Given this input and training process, AVENUE transfer rules are designed to use syntactic transfer that is guided and augmented with feature structures. The source sentence analysis and word-alignments are sufficient to learn this type of rule. Transfer could be done purely syntactically if little feature information was available, with feature constraints being added as available to decrease ambiguity and improve translation quality.

As described above, the transfer rules embody all three stages of trans-

lation: analysis, transfer and generation. Rules do not enforce the use of a specific computer character set (such as ASCII or ISO-8859-1), but transfer rules between two languages with different orthographies would likely use the UTF-8 encoding of Unicode (Unicode Consortium, 1996). Unicode is able to handle most major and many minor world languages, and also has space to add in new scripts, which may be necessary when dealing with the languages the AVENUE project is targeting.

All rules have a unique identifier, usually the name of the top-level source constituent (such as S, NP, VP, etc.) plus an integer index into all rules with that constituent name. In the example transfer rule shown in Figure 3.1, the unique identifier is `{S,1}`. Following the identifier are the production rules for the source and target sides, with the source production rule used in analysis and the target in transfer and generation. In the example rule, this line is "S::S : [NP VP] -> [VP NP]". The first S refers to the left-hand side of the source production rule and its feature structure is referred to as X0 elsewhere in the rule. After the "::", the second S refers to the type of target constituent the source constituent will be mapped to during transfer. Its feature structure is referred to as Y0. Even though the source and target left-hand sides are identical ("S") in this example, they can be different.

```
{S,1} ; Unique identifier
S::S : [NP VP] -> [VP NP]
; Source and target production rules
(
  ; Constituent alignments
  (X1::Y2)
  (X2::Y1)

  ; Analysis unification equations
  ((x0 subj) == x1)
  (x0 = x2)

  ; Transfer Equations
  (y0 = x0)

  ; Generation equations
  (y2 == (y0 subj))
  (y1 = y0)

  ; Feature completion
  ((y0 agr) = (y1 agr))

  ; Generation Constraints
  ((y1 agr) = (y2 agr))
)
```

Figure 3.1: Sample Transfer Rule

For example, a source language NP could be mapped to a PP in the target language.

Following the top constituent labels are the source and target language production rules. In this case the source S produces an NP and VP. Later

in the rule, references to these source constituents are done with X followed by the constituent index, starting at one. So NP is referenced as X1 and VP is referenced as X2. On the target side the production rule order is VP and NP, referred to with Y1 and Y2, respectively.

The body of the rule begins with one or more constituent alignments that describe which constituents on the source side should transfer to which constituents on the target side. In this example the alignment indicators are (X1::Y2) and (X2::Y1), showing that the NPs and VPs on both sides align with each other, though in reverse order. Not all constituents need to have an alignment to the other side, allowing for translation cases where a source language constituent is not needed in the target language. Similarly, a target constituent that is not aligned to a source constituent entails that a constituent will be inserted based upon the feature structure assigned to it in the rule's unification equations (described below). In addition, a double-quote enclosed string may also be used in the target side, directing the transfer engine to always insert this exact word into the translation in relative position to the rule's other target constituents. An example of word insertion is detailed in the next chapter.

After the constituent alignments for the rule are a set of feature uni-

fication equations. Unification equations are used in each of the analysis, transfer, and generation stages. The feature unification equations used in the rules follow the formalism from the Generalized LR Parser/Compiler (Tomita, 1988) and the following is adapted from Tomita's description. The actual unification is done using the UKernel library, Benjamin Han's C++ port of the original Tomita unification engine (Han, 2001). The transfer engine uses unification equations which "are very similar to [Lexical Functional Grammar] equations except that they use the variables x0, x1, x2, x3, etc. in place of the up-arrow and down-arrow" (Tomita, 1988). The AVENUE transfer rule formalism extends these equations to have Xn refer to source language constituents and Yn refer to target language constituents.

The left hand side of an equation is a path. A path is:

- A variable (e.g. x0, x1, y0, y1, etc.).
- A variable followed by any number of character strings separated by spaces (e.g. (x1 subj), (x0 agreement), (x2 xcomp subject), (y3 number)). When used in UTF-8 mode, the character strings themselves can be any valid non-space or punctuation string available in Unicode. This type of path must be enclosed in parentheses.

The right hand side of an equation is:

- A path.
- A character string (e.g. foot, tóutòng, $\alpha\beta\gamma$, 12, *M) excluding some special characters such as the quotation mark but including most alphanumeric glyphs from Unicode when UTF-8 mode is used.
- A list consisting of the word *OR* followed by any number of character strings (e.g. (*OR* nominative accusative), (*OR* sg pl)).

Each equation is enclosed in parentheses. The unification operator, the equal sign '=', actually does pseudo-unification. In full unification, both feature structures on the right and left-hand side are completely unified where a reference to one is the same as a reference to the other. However, in pseudo-unification only the left-side path is modified to reflect unification. The right-side feature structure is left unchanged.

Examples:

(x0 = x1) x1's feature structure is unified with x0's, with the unified structure copied into x0 if unifiable.

((x0 subj) = x1) Copy x1's feature structure as the value of x0's subj feature.

((y0 case) = (*OR* nominative accusative)) Unify if the value of case in y0 is empty or has a value of either 'nominative' or 'accusative'.

((x1 agreement) = (x2 agreement)) Attempts to unify x1's agreement with x2's agreement. Succeeds if either x1 and x2 have the same value for agreement or if either x1 or x2's agreement is empty. Fails to unify if x1 and x2 both have a value for agreement and the values differ.

((y2 root) = be) If the value of root in y2 is empty, set it to 'be', if non-empty, succeed if root's value is 'be', fail otherwise.

In addition to the pseudo-unification operator '=', several other equations are supported:

Constraint equations '=c' Same as '=' but constrains the right and left hand-side values to be non-empty. Both values must exist and be equal, e.g. ((x1 case) =c nominative).

Assign and Remove equations '==' The right-side path is unified with the left, but then the value of the right-side path is deleted, e.g. (y2 == (y0 subj)).

Overwrite equations '<=' The value of the left-side path is overwritten by the value from the right-side. No unification occurs.

Negative equations '*NOT*' The value of the left-side path must be anything but the right-side value, e.g. ((x2 subcat) = (*NOT* intransitive)).

Existence equations '*DEFINED*' and '*UNDEFINED*' Checks to see if the left-side path does or does not have a value, e.g. ((y1 negation) = *UNDEFINED*) only succeeds if (y1 negation) has no value.

Transfer unification equations are generally divided into three types: x-side constraints, where paths only refer to source language constituents and which are used during analysis; x-y side constraints, used during transfer to selectively move feature structures from the source to the target language structures; and y-side constraints, used during generation to distribute target language feature structures and to check agreement constraints between target constituents.

3.2 Analysis

Analysis is the first stage of the translation process. This stage includes morphological processing and unification-based syntactic parsing. Unlike systems such as METAL and Eurotra, morphological analysis is not a separate stage

of analysis but is integrated into the parse. The morphological analysis will only occur if the appropriate morphology data files are specified for inclusion, otherwise the words are always looked up directly in the lexicon.

Parsing itself is done bottom up using an unification-based chart parser, with additional enhancements to avoid exploring impossible paths. Before any parsing the system optionally normalizes the case of the input text. Case normalization can be done over multilingual text when in UTF-8 mode. Other systems described in the previous chapter do not mention having this ability. Word tokenization is done by using whitespace and punctuation as word boundaries.

During parsing, each word is looked up in the transfer lexicon and, if the word is not found and morphological analysis is enabled, run through a morphological analysis. Like the Corelli system, this analysis can return more than one word stem and feature structure for the word, reflecting the possibility that a word could be analyzed multiple ways. The morphological analysis itself is currently done using a version of PC-KIMMO (Koskienniemi, 1983) modified by Christian Monson, who is researching morphology analysis and generation for AVENUE. Use of PC-KIMMO as the morphology component will likely change as the system matures and more morphology research is

done, with the eventual goal that this morphological analysis itself, like the transfer rules, would be automatically learned. After morphological analysis, the returned stem is retrieved from the combined analysis/transfer lexicon.

Chart parsing results are stored using two data structures: a vector of constituents (which doubles as the chart) and a vector of arcs. A constituent is a one word or longer phrase of some grammatical type, from parts of speech to phrases such as nouns or verb phrases, all the way up to the sentence level. A constituent stores the beginning and end indices of the words it covers, the constituent name (e.g. S, NP, VP, ADV, etc.), and the arcs representing parses that were used to construct the constituent. A single constituent could have multiple underlying parses that led to that constituent type over that sequence of words. These parses and the rules that led to them are stored in the vector of arcs. An arc stores the index of the analysis rule that created it, the start and end positions of the words covered by the arc, the constituents that were completed to make the arc, and for uncompleted arcs, the number of completed constituents.

So, for example, in the sample analysis tree in Figure 3.2, the largest constituent is "S", covering the whole sentence of words 1 to 3. It has a pointer index into the array of arcs. That arc indicates what analysis rule

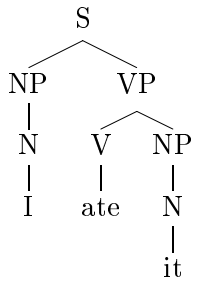


Figure 3.2: Sample Tree

was used to form the constituent (`S -> NP VP` in this case which might be rule 1). The arc in turn has links back into the constituent array for the NP and VP, which would then have links back into the array of arcs, and so forth, to the leaves of the tree representing words. This is how a parse tree is represented inside the transfer engine.

Using this method, multiple possible parses of the same input can be stored together. This ambiguity packing is done by allowing a constituent to have pointers to multiple arcs, each representing one possible parse. Unlike the Corelli system, constituents in the chart that are subsumed by larger constituents are not discarded. I do not believe that this is a major performance hit and moreover it is needed to do partial translations of the sentence when a full parse is not possible. In addition to packing normal syntactic ambiguities in the structure of the source sentence, the combination of analysis

and transfer in the rules means the parser will also pack two rules where the parse would be identical, but the transfer to the target language would be different.

The constituent type that the source text should parse to can be set to any value, meaning that the transfer engine can be used to translate whole sentences or smaller constituents such as noun or verb phrases. Initial system testing has been on noun phrases. The end result of the analysis is the ambiguity-packed chart of constituents, along with a set of arcs detailing how the constituents relate to each other and the rules from which they were derived. Each possible parse tree, a unique combination of constituents and arcs, is also associated with a feature structure built up by the x-side unification equations in the rules.

3.3 Transfer and Generation

Analysis creates a set of parse trees and associated feature structures for the input text, with each tree representing a potential translation. Each branch of a parse tree is associated with a transfer rule, since as explained above, each analysis rule also includes a transfer component. During transfer, the

```
; handles phrases such as 'in the house' or 'on the table'
{PP,1}
PP::PP : [PREP NP] -> ["zai4" NP PREP]
(
(X1::Y3) ; switch the order of the NP and PREP
(X2::Y2)
((x0 obj) = x2) ; build the source f-structure
(x0 = x1)

(y0 = x0)

((y0 loc) =c +) ; check that
(y2 == (y0 obj))
(y3 = y0)

)
```

Figure 3.3: Transfer rule for English to Chinese locative prepositional phrases

system works through these trees in succession until one can be successfully translated. Translating the tree involves a walk through the chart of arcs and constituents created during parsing. Counters in each constituent keep track of which arcs have been visited and which have not. In this way, the system keeps track of which trees have and have not had transfer attempts. Transfer and generation in the AVENUE translation engine are interleaved, a consequence of the trained nature of the rules, reflecting the fact that a separate generation grammar is not available for the target language.

Transfer creates a set of constituents and arcs similar to the ones cre-

ated during analysis, except that they represent the target language syntax tree. Transfer starts at the top constituent in the source representation and creates a top constituent for the target language tree. The x-y-side transfer unification equations in the rule in the current arc for the constituent are then applied to create a feature structure for the top target constituent. Features and values can be passed from the source to the target f-structure, or be created in the target f-structure during transfer. In the transfer rule in Figure 3.3, which handles the transfer of locative prepositional phrases from English to Chinese, feature transfer is done with the simple equation "(y0 = x0)".

The equations can also check for constraints on certain values that are needed for the rule to apply. In the Figure 3.3, the rule only applies if the prepositional phrase is locative which is enforced by "((y0 loc) =c +)". If any of the transfer equations fail to unify, the transfer attempt stops and the next possible tree is explored. Unification equations can also pass features to the children of the current target constituent, as in the example from Figure 3.3, where the target NP get the value of the PP's object with the equation "(y2 == (y0 obj))".

After creating the top target constituent, transfer and generation are

then run recursively on the child constituents in an order determined by the x and y-side constituent alignments listed in the rule. For the rule in Figure 3.3, this means that after transferring the PP, it next inserts the word *zai* into the target tree, then transfers the source NP and then the PREP, using the order indicated by the constituent alignments. The transfer rule for the NP would be run on the NP (and recursively on its children) and then on the PREP. Another example would be if the sentence object NP came last in English but first in another language. This would be reflected in the constituent alignments. The object constituent would consequently be explored and transferred next, with the subject and verb coming after.

Again, if at any time any of the unification rules failed, the transfer attempt is halted and the system updates what paths have been explored so far. The system then tries the next tree representing a different path through the chart. This continues down to the lexical level until a complete target tree has been created.

Transfer at the lexical level is also done with feature unification. Features passed down to the lexical level are checked against the unification equations of lexical transfer rules whose source side matches the source word or word stem. If unification is successful the corresponding target word is inserted

```
N::N |: [wo3] -> [I]
(
(x1::y1)
((x0 form) = wo3)
((x0 py) = wo3)

((y0 case) = nom)
((y0 agr) = 1sg)
)

N::N |: [wo3] -> [me]
(
(x1::y1)
((x0 form) = wo3)

((y0 case) = acc)

)
```

Figure 3.4: Sample lexical rules

into the target tree structure along with the results of the lexical transfer unification. Figure 3.4 shows two lexical transfer rules where one Chinese word is translated in two different ways. During parsing, the Chinese word (represented by its pinyin Romanization "wo3") would be tagged as having nominative or accusative case. This case feature is then passed down during transfer and generation until it reaches the lexical level. At that point the target word with the appropriate case is selected. No generational morphol-

ogy is currently done in the system, though it will be added later.

3.4 Generation Constraint Checking

Once the transfer stage has found a candidate target tree, the system needs to check the tree before producing a final translation. In many cases, the target language will enforce constraints that are not found in the source language. One such example happens when translating from Chinese to English. Chinese does not have agreement between the subject and the verb of a sentence. The verb form remains unchanged for all subjects. However, in English subject verb agreement must be enforced. Since full unification is not used, and the transfer pass goes from the top constituent down to the lexical level and not bottom up, the necessary agreement information at the sentence level is not available until the target word translations have been selected.

For example, agreement information is available in the lexical rules shown in Figure 3.5, but the constraint check is done higher up in the target tree with the rule in Figure 3.1 that has the equation "((y1 agr) = (y2 agr))". In order to resolve this paradox, a bottom-up "constraint check/feature filling"

```
V::V |: [chi1] -> [eat]
(
(x1::y1)
((x0 form) = chi1)

((y0 tense) = pres)
((y0 agr) = (*not* 3sg))
)

V::V |: [chi1] -> [eats]
(
(x1::y1)
((x0 form) = chi1)

((y0 tense) = pres)
((y0 agr) = 3sg)
)
```

Figure 3.5: Sample lexical rules used in constraint checking

pass is run.

Starting at the lexical level of the tree, the engine runs the constraint checking equations that pertain to the lexical items. Once the constraints are run, the engine will then run equations that pass feature values from lexical nodes up to the next higher level the generated target tree. For example, in the example rule in Figure 3.6, the unification equation "((y0 agr) = (y1 agr))" passes the agreement value from the verb to the verb phrase, where it is checked in the rule in Figure 3.1. Once the next highest level of the

tree has all the features needed for the constraint checks from lower in the tree, then constraints equations for that level are run, and feature values are passed up yet again. This is done until the top sentence level is reached.

```
; Transfers transitive verb constructions
{VP,1}
VP::VP : [V NP] -> [V NP]
(
  (x1::y1)
  (x2::y2)
  ((x2 case) = acc)
  ((x0 obj) = x2)
  (x0 = x1)

  (y0 = x0)

  ; Forces use of 'him' and 'her' instead of 'he' and 'she'
  ((y2 case) = (y0 obj case))

  ; Pass up agreement feature to use in constraint checks
  ((y0 agr) = (y1 agr))
)
```

Figure 3.6: Sample Transfer Rule Showing Constraint Checking

A rule can also insert a target constituent that is not aligned with a source constituent. Currently, only lexical-level constituents may be added. This insertion is done when the system detects a target side constituent that is not aligned in the rule with a source constituent. The unification equations for the rule should add in feature value pairs to be stored in the constituent. During lexical transfer, lexical transfer entries for matching that constituent

type are searched and their unification rules run against the feature structure already created for the constituent. The target word from the first lexical transfer entry that successfully unifies is then used as the value of the inserted word.

Once all phases have completed successfully, the lexical leaves of the target language tree are read in a depth-first search, concatenated into a string, and returned. The system can either stop at the first translation found or continue to find all translations allowed by the translation rules. Other systems mentioned in chapter 2 only give one possible translation. The benefit of being able to return multiple possible translations is that they can be input to another program, such as a multi-engine MT system, which would then choose the best translation. The system also returns the feature structure for the entire target sentence along with the feature structures for each individual translated word.

3.5 Partial Translations

With the complexities of language and the difficulty of parsing long sentences, the transfer engine will be unable to find a complete parse for many input

sentences. Even if a complete parse were available, the system will not always be able to transfer that parse. Even in these cases, the AVENUE system is still able to produce partial translations. When a complete parse or transfer is not available, the system searches for the largest constituents for which a parse is available and attempts to transfer these. The system will search for the set of the largest constituents that cover the source sentence. If a source word is not part of one of these constituents, the system reduces to direct translation by just translating the word. In the worst case, the entire sentence will be translated word by word. When a word is not in the transfer lexicon, the original source word is inserted in the translation. The system also has facilities for doing number translations, since it would be impossible to store all possible numbers in a transfer lexicon. These number translations are done with special functions designed to understand and produce numbers of the source and target languages. With these partial translation methods, even if a fluent translation cannot be produced, at least the general meaning of the source sentence can be inferred.

3.6 Training Adaptations

Several specific functions were added so that the transfer engine could be incorporated into the training process. The first of these was a way to dynamically add rules into the system. Also added were ways of turning rules off for a run of the system. Another unique feature is the ability to turn off all rules but one for a specific type of constituent. In this way, the training component can see how that particular rule affected the translation outcome. Depending on the result, the training component will replace the rule with a more general rule that could handle a broader range of similar structures.

The transfer engine also needs to be able to return detailed information about the results of the translation. Since an important part of the training involves word-aligned phrases or sentences, it is necessary to be able to return the resulting feature structure for each translated word, to see what features were eventually passed down to the lexical level. The final feature structure for the translated sentence as a whole also needed to be stored and returned. The training program can utilize all this information to refine the transfer rules.

Chapter 4

Linguistic Coverage and Examples

4.1 Translation Test Cases

In Trujillo (1999), the author discusses several problems that are particular to machine translation. The problems result from the "divergences and mismatches between source and target sentences", where a divergence is defined as case where "the meaning is conveyed by the translation, although syntactic structure and semantic distribution of meaning components is different in the two languages." The following section details these various divergences

and lists example transfer rules from various language pairs to either show how the AVENUE transfer engine can handle them or what work needs to be done for the engine to handle them. The Chinese rules come from a working system, while the the rules for other languages are possibly incomplete, but sufficient to demonstrate the concept.

While all the divergences can be handled by the current formalism, the ability of each transfer rule to only act on constituents one-level down in the tree may mean that rules would need to produce shallower trees than might otherwise be necessary. An example of this is the thematic divergence example below, where the need to shift the English object to the front of the Spanish sentence leads to a rule with the object NP exposed like at the sentence level: `S -> NP V NP` as opposed to `S -> NP VP`. While the ability to search and manipulate deeper levels of the tree would be useful, it could be difficult to learn these kinds of rules given the current training method.

4.1.1 Thematic Divergence

According to Trujillo, "thematic changes relate to changes in the grammatical role played by arguments in a predicate." An example of this from Spanish is:

Ella *te* *gusta.*
She you-ACC pleases
You like her.

In this case, the subject and object switch in the Spanish translation. This particular translation could be accomplished with this rule in the AVENUE formalism:

```
{S,1}
S::S : [NP V NP] -> [NP NP V]
(
  (x1::y2) ; move the subject to object position
  (x2::y3) ; move the verb to the end
  (x3::y1) ; move the object to subject position

  ((x0 subj) = x1) ; build source f-structure
  ((x0 obj) = x3)
  (x0 = x2)

  (y0 = x0)
  ((y0 root) =c like) ; check to make sure the rule applies
  ((y0 subj) <= (x0 obj)) ; transfer feature structures
  ((y0 obj) <= (x0 subj)
  ((y0 subj case) <= nominative) ; set appropriate argument case
  ((y0 obj case) <= accusative)

  (y1 == (y0 subj)) ; distribute f-structures
  (y2 == (y0 obj))
  (y3 = y0)

  ((y1 agr) = (y3 agr)) ; enforce agreement
)
```

In the above rule, the positions of the subject and object are rearranged according to the Spanish ordering. The rule only applies to sentences whose

root (which would be the verb infinitive form) is 'like'. The case of the arguments is set appropriately and the feature structures are distributed to the noun phrases. Agreement is enforced between the Spanish subject and verb.

4.1.2 Head Switching

In head switching, a word that was the head of the phrase (such as a noun or prepositional phrase) is no longer the head of the translated phrase, becoming a modifier or complement. The head is instead replaced by some other part of the phrase.

The example comes from English to German (Hutchins & Somers, 1992), in sentences where one expresses liking some activity, such as the following:

Ich *schwimme* *gern.*
I swim gladly
I like swimming.

The head of the verb phrase in English is the verb 'like' with a dependent participal complement ('swimming' in this case'). In German 'like' instead becomes the adverb 'gern' and the main verb is replaced by the verb form of the participal complement.

This could be handled by a verb phrase level rule:

```
{VP,1}
VP::VP : [V V] -> [V ADV]
(
  (x1::y2)
  (x2::y1)

  ((x2 vform) =c participal)
  ((x0 xcomp) = x2)
  (x0 = x1)

  (y1 == (x0 xcomp)) ; transfer f-structure
  ((y1 root) =c like) ; constrain application of rule
  (y2 = x0)
  ((y1 vform) <= finite) ; make sure German main verb is finite

  ((y0 agr) = (y1 agr)) ; pass agreement up for sentence agreement check
)
```

In the rule the English verb and complement are switched in the German translation. The rule is constrained to only apply when the English verb is 'like'. The verb form of the former participal is set to finite.

4.1.3 Structural Divergence

Structural divergence occurs when a phrase in one language is expressed as a phrase of a different type (e.g. PP instead of NP) in the target language. For example, in English 'enter' takes an NP as an argument, while the Spanish equivalent takes a PP (Trujillo, 1999).

Luisa entró a la casa.
Luisa entered to the house.
Luisa entered the house.

This type of divergence could be handled with the following rule:

```
{VP,6}
VP::VP : [V NP] -> [V ‘‘a’’ NP]
(
  (x1::y1)
  (x2::y2)

  ((x0 obj) = x2)
  (x0 = x1)

  (y1 = x1)
  ((y1 root) =c enter)
  (y3 = x2)

  ((y0 agr) = (y1 agr))

)
```

4.1.4 Lexical Gap

With a lexical gap, a concept which requires several words in one language is expressed with only one word in the target language, or vice versa. The MT system could try to translate the multi-word phrase compositionally which seems to be the approach taken by the Systran system available at http://babelfish.altavista.com for several test sentences. The system could

also have rules to identify the concept and either condense or expand it as necessary to match the target language.

English and Chinese have a systematic lexical gap for prepositions indicating location ("in", "on", etc.) which are expressed with two words in Chinese.

Wo3 *zai4* *fang1jian1* *li3.*
I at room inside
I am in the room.

A rule going from English to Chinese would need to recast the preposition as two separate words:

```
{PP,3}
PP::PP : [PREP NP] -> ["zai4" NP PREP]
(
(X1::Y3)
(X2::Y2)
((x0 obj) = x2) ; create source f-structure
(x0 = x1)

(y0 = x0) ; feature transfer

((y0 loc) =c +) ; check that the prep is a locative preposition
(y2 == (y0 obj)) ; distribute features
(y3 = y0)
)
```

Here the system first builds the English prepositional phrase, but checks to make sure the PP deals with location (as opposed to direction) and introduces the word *zai4* ('at') in addition to moving the preposition at the end of

the phrase, where it will be translated with the appropriate transfer lexical entry.

4.1.5 Lexicalization

Different languages may distribute semantic content differently among words in a phrase. For example, with expressions of movement in English "it is the model of transport that is expressed by the verb ... and the direction is expressed by prepositions. In ... French, it is the direction which is expressed by the verb ... and the mode of transport is expressed by adverbial adjuncts" (Hutchins & Somers 1992).

Il traversa la rue á pied
He crossed the road on foot
He walked across the road

For these French examples, it would be hard to generalize for all movement verbs, so a rule for each kind would be included. Here is an example rule from English to French for the above example:

```
{VP,5}
VP::VP : [V P NP] -> [V NP ``a pied'']
(
  ; x1 is replaced with 'a pied', no alignment needed
  (x2::y1)
  (x3::y2)
```

```
  ((x0 direction) = x2)
  ((x0 direction object) = x3)
  (x0 = x1)

  (y0 = x0)
  ((y0 root) =c walk) ; only applied for 'walk'
  (y1 = x2)
  (y2 = x3)

  ((y0 agr) = (y1 agr))
)
```

In the above rule, the preposition becomes the verb in the French sentence. Since each verb of motion has a different adverbial adjunct expressing the means of motion, this is inserted directly in a separate rule for each movement verb. In the example rule above, application is constrained to the case where the original English verb is walk. The destination NP is translated by other NP rules.

4.1.6 Categorial Divergence

When a word can be translated by a word that is semantically similar but has a different syntactic category, this is an case of categorial convergence. One example of this between Chinese and English is the way that adjectives are translated.

In Chinese, adjectives in the sentence's predicate that describe the subject

do not need a copula, as adjectives do in English with "be". Instead, in Chinese they act as verbs themselves and are called "stative verbs". The stative verbs are commonly used with the adverb for 'very'. For example:

Wo3 hen3 gao1.
I very tall
I am tall.

Below is a rule for this situation:

```
{VP,3}
VP::VP : [VS] -> [V ADJ] ; VS for ‘‘stative verb’’
(
  (X1::Y2) ; align stative verb with adjective
  (x0 = x1)

  (y0 = x0) ; transfer feature structure

  ((y1 form) = be) ; introduce copula

  ((y0 agr) = (y1 agr)) ; pass agreement to VP to check for subject agreement
)
```

4.1.7 Collocational Divergence

According to Trujillo, "collocational divergences arise when the modifier, complement or head of a word is different from its default translation". This can be handled in the rules by having several versions of a rule with the first few checking for conditions where the word would not use the default

translation, and the last rule being the default translation. This would differ from the multi-lexeme divergence below in that the translation would still be compositional.

For example, in English the normal sense of "have" is to own something, but in the phrase "have some dinner"it means 'to eat'. 'Have' could not be translated with the same word for both senses in Chinese.

Wo3 you3 yi1ge5 dian4hua4.
I have a telephone
I have a telephone.

Ni3 ying1gai1 chi1 yi1dian3 wan3fan4.
you should eat a little dinner
You should have some dinner.

To solve this problem, we could add multiple versions of the source word in the transfer lexicon, with an unconstrained default as the final choice. Other choices for translating the verb have constraints based on the semantic category of the verb phrase's object.

For example:

```
VP::VP : [V NP] -> [V NP]
(
  (X1::Y1)
  (X2::Y2)
  ((x0 obj) = x2)
  (x0 = x1)
```

```
  (y0 = x0)

  (y2 == (y0 obj))
  (y1 = y0)
  ((y1 object semtype) = (y2 semtype))
)

V::V |: [have] -> [chi1]  ; meaning eat
(
 ((x0 form) = have)

 ((y0 object semtype) =c food) ; constraint on object type
 ((y0 form) = chi1)
)

V::V |: [have] -> [you3]  ; meaning possess
(
 ((x0 form) = have)

 ; no constraint on object type
 ((y0 form) = chi1)
)

N::N |: [dinner] -> [wan3fan4]
(
 ((x0 semtype) = food)
 ((x0 form) = dinner)

 ((y0 form) = wan3fan4)
)
```

4.1.8 Multi-lexeme and Idiomatic

In multi-lexeme and idiomatic divergences, a phrase in one language whose meaning in non-compositional is translated by a similarly non-compositional

phrase in another language. This is currently not well supported in the AVENUE transfer engine since analysis only works with single words and cannot deal yet with compounds. For a compound or idiom that parses to a phrase, it could be possible to identify the compound through checking inside the phrase's feature structure. An appropriate transfer could then be attempted. See the section in "Future Work" for further discussion.

4.2 Chinese to English Example

To illustrate the transfer algorithm, I will describe the translation of a Chinese sentence into English, using a small Chinese to English transfer grammar included in Appendix A. Rule names mentioned here refer to rules in this sample grammar.

The example sentence demonstates some of the abilities of the grammar to do appropriate target lexical selection, to insert new words, both directly as a string or based upon a feature structure created by the transfer process, and to remove source language constituents that are not directly translated. The example sentence is the following question in Chinese (written using the pinyin Romanization system):

Example sentence:

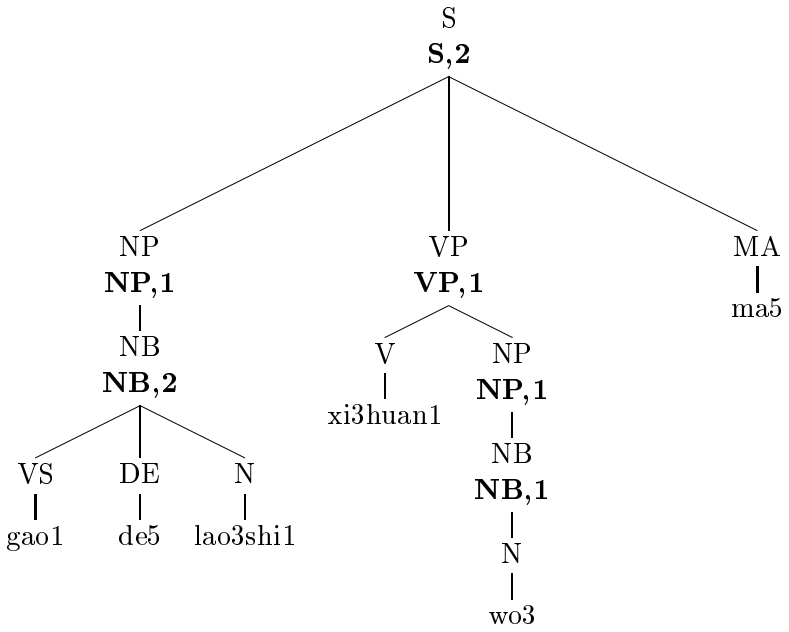
Gao1	*de5*	*lao3shi1*	*xi3huan5*	*wo3*	*ma5*
tall	MOD	teacher	like	1sg	QUEST

Does the tall teacher like me ?

De5 is a particle used with modifiers in Chinese noun phrases and *ma5* is a particle used at the end of sentence to indicate it is a question.

Analysis of the source sentence produces several possible parse trees reflecting the several ways transfer could precede. Currently the order the trees are explored is determined by the order in which the rules used in the trees appear in the transfer rule text file.

The first parse tree and feature structure explored are listed below, where under each tree node is the name of the rule in bold that produced that branch:



Feature Structure:

```
((subj
 ((form lao3shi1)
  (py lao3shi1)
  (mod
    ((form gao1 )))
  (case nom)))
 (obj
  ((form wo3 )
   (py wo3)
   (case acc)))
 (form xi3huan5 )
 (py xi3huan1)
 (act question))
```

Since transfer always starts at the top of the tree, the first transfer rule fired is `{S,2}`. First the transfer equations in `{S,2}` are run: `(y0 = x0)` and `((y0 act) =c quest)`. The source f-structure is passed to the new

target S node, then the value of `act` (short for "speech act") is checked. The check succeeds since the value is "question". After the transfer equations run successfully the new top target tree constituent is created and the transferred feature structure is assigned to it.

Next the system runs generation unification equations to distribute and check features from the root constituent to the child constituents (AUX NP VP in this case). `{S,2}`'s equations are `((y1 form) = do)` , `(y2 == (y0 subj))`, and `(y3 = y0)`. Y2 (the NP) gets the subject f-structure, Y1 (an inserted constituent) is assigned 'do' as its base form to be used during lexical selection, and Y3 (the VP) is assigned everything else. Next target constituents are created for the children and are assigned the feature structures created during the previous step. Transfer then proceeds recursively according to the rule's source and target constituent alignments. In this case, the unaligned inserted constituent AUX is explored next.

Since the AUX is not aligned with with a source constituent, the system tries to find its surface form by searching all the AUX lexical transfer entries, stopping at the one whose equations successfully unify with AUX's f-structure. The system finds the entry for 'do' whose form feature unifies with AUX's, additionally adding in the agreement value of (*or* 1sg 2sg pl).

Note that this entry will cause this transfer attempt to fail later because the agreement will not agree with the subject's agreement of 3sg.

Since there is nothing to explore under the AUX, the system next returns to the NP produced by rule `{NP,1}`. The transfer unification equation `(y0 = x0)` and the generation equation `(y2 = y0)` are then applied. Next the target child constituents are created: the string "the" is inserted into the target tree and the NB constituent is created. The "the" is inserted because Chinese does not use articles for common nouns, so one must be inserted for English.

Transfer continues on to the NB constituent using transfer rule `{NB,2}`. This handles nouns modified by adjective-like stative verbs. The f-structure is transferred over, and the case and num(ber) features are passed down to the noun N.

Chinese uses a particle *de5* after the stative verb. No equivalent exists in English for *de5*, so it does not have an alignment. The stative verb aligns with the English adjective ADJ while the Chinese noun aligns to the English noun N. Both of these words are then lexically transferred to 'tall' and 'teacher', respectively.

With the Chinese subject NP fully transferred, transfer continues on the

VP from rule `{VP,1}`, which handles transitive verb constructions. The VP f-structure is transferred and the English NP is assigned the case from the object (accusative in this instance). Transfer proceeds to the children. V is lexically transferred to 'like' (with the verb form set to infinitive) and the NP is transferred by rule `{NP,1}`.

In `{NP,1}` the f-structure is transfered, then the generation equation passes the f-structure down to the target NB. Transfer continues on NP's children. "The" is inserted into the target tree and transfer occurs on the NB with rule `{NB,1}`. `{NB,1}` transfers the f-structure over, while the case and num features are passed to the noun N during generation. Lexical transfer then occurs on the noun "wo3". The system tries the first lexical transfer entry for "wo3" to "I". However, "wo3" was assigned accusative case during analysis while "I" is constrained to nominative case. Unification fails and the transfer attempt fails.

Since transfer involved the insertion of a new constituent based on feature structure, the engine currently makes another attempt trying the same tree but assuming the failure was due to picking the wrong word for the insertion. It tries the next word, "does" in this case, that could be inserted. Since this was not the reason for the failure of the last attempt, this attempt will also

fail. This is an inefficiency that will be corrected in future versions of the transfer engine.

In the next attempt, the system tries the next lexicon entry for "wo3" which is "me" and succeeds in the lexical transfer. At this point a full target tree has been created (using "do" as the inserted AUX) and the y-side constraints must be checked. First, the system runs a "feature-filling" pass to allow features in the target lexical f-structures to be passed up the tree. For example, agreement and noun type information is passed up by the NP and NB rules. Noun type could be common, pronoun, or proper. Agreement and verb form (finite or infinitive) information are passed up the VP rules. With the target f-structures fully filled, the generation constraint equations are then run. In this case the main constraints are in `{S,2}` where the verb phrase verb form must be inf for infinitive and the NP and VP must have the same agreement. In this case since "do"'s agreement is `(*or* 1sg 2sg pl)`, this agreement check will fail and so will the transfer attempt.

In the next attempt the correct translation for "wo3" and the correct form of "do" are chosen. However, this particular tree used `{NP,1}` to transfer the object "wo3". This rule is constrained to apply only for common nouns not pronouns, since it inserts "the". The generation constraint in `{NP,1}` thus

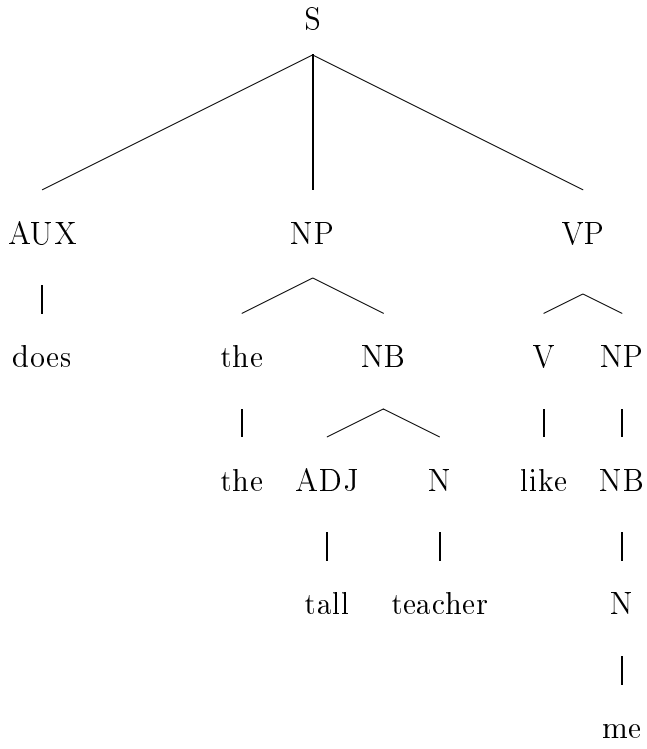
fails and the next transfer attempt is made.

The next transfer attempt uses a different analysis tree than used above, with `{NP,2}` instead of `{NP,1}` for the sentence object NP. With this tree, the system agains works its way through the possibilities until it find the correct form of "do" and the correct translation for "wo3". The object NP is constrained to be a common noun so that also succeeds. Since the unification equations for all the rules in transfer, generation, and agreement checking succeeded, the system considers this a valid parse. It concludes by reading the lexical leaves of the target tree to produce: "does the tall teacher like me ?".

The transfer rule application order for the successful transfer was:

```
**** TRANSFER ATTEMPT #11
*** TRANSFER RULE #1: S,2
*** TRANSFER RULE #2: NP,1
*** TRANSFER RULE #6: NB,2
*** TRANSFER RULE #7: VP,1
*** TRANSFER RULE #3: NP,2
*** TRANSFER RULE #5: NB,1
```

The final generated English tree is:



If the transfer engine were set to only find one translation, it would return this one. Note that there is one other possible translation "does tall teacher like me ?" which is allowed because the first noun phrase could also be translated by `{NP,2}`, which does not have a restriction on not applying to common nouns, though one could be added. In this case the first "the" would not be added. The system can be set to return as many translations as are possible given the transfer rules.

Chapter 5

Results

An important achievement of the transfer engine has been its integration and use with the rule learning system currently also being developed by Kathrin Probst as part of the AVENUE project (Probst, 2002). Simple German to English noun phrase rules have already been successfully learned from word-aligned parallel text. When compositional rules can be learned, the system is ready to support them.

To test the system's usefulness in rapid development machine translation, I conducted an experiment over the course of one week where I developed a small translation system. I manually wrote a small set of rules for translating Chinese to English. These rules demonstrate the system's ability to handle a variety of translation divergences. Among the system's current abilities are target word insertion based on f-structures, constituent reordering, genera-

tion constraint checking, and proper target lexical selection. Examples and descriptions of these can be found in the previous section.

The translation lexicon was automatically generated using the Chinese to English glossary available from the Linguistic Data Consortium of the University of Pennsylvania. This glossary has 54,000 Chinese entries each with one or more English definitions and glossaries, but not part of speech information. To supply part of speech information for the lexical transfer rules, I used the English word part of speech data frequency list derived from the British National Corpus (Leech, 1994). The parts of speech used included detailed information such as tense and agreement for verbs and number and countability (mass vs. count) for nouns. This extra information was incorporated as feature constraints in the lexical transfer rules. When the part of speech of the English gloss was unknown, the lexical transfer entry used UNK for unknown. A transfer lexicon with approximately 30,000 entries was generated in this manner, though the automatic method used means many entries are spurious or have incorrect part of speech and feature constraints.

Testing was done over a set of 993 Chinese sentences, many of which are over fifteen words long. The sentences were not used during development

and were only inspected to determine the cause of system crashes during translation. Most of the sentences are drawn from Xinhua newswire from the People's Republic of China. Scoring was done through an automatic MT scoring algorithm from the IBM Thomas J. Watson Research Center. This method, called the Bleu score, compares the output of an MT system with several human-produced reference translations. Based upon the number of word segments that are found to correspond between the MT output and the references, a score is assigned (Papineni, 2001). The more word sequence matches found the higher the score. Scores can range from 0 to 1, with a score of 1 reserved for a translation that perfectly matched one of the reference translations. Scores on human translations have ranged from 0.2 to 0.3, so a score does not have to be near 1 to indicate a good translation.

For comparison I used a Chinese to English multi-engine MT system currently running at the Language Technologies Institute (Zhang, 2001). This MEMT system takes input from a example-based MT system and a glossary-based MT system, but can have difficulty with constructions where Chinese and English word order differ. The best Bleu score for the MEMT system on the test set of 993 Chinese sentences is currently 0.1605. This score reflects nearly two years of development work on the MEMT system. After only one

week of development I was able to produce a system whose score is 0.0749, nearly half the score of the MEMT system but with much less development time and effort. This demonstrates the AVENUE transfer engine's usefulness for rapid development of machine translation systems. Combined with the training component of AVENUE to automatically learn rules, the transfer engine will enable faster language pair translation development than for current transfer-based MT systems.

Chapter 6

Future Work

6.1 Rule Selection

Currently the system tries transfer rules in the order in which they are loaded into the system. Consequently, it is best to put more specific rules first followed by more general rules that could act as defaults if none of the others succeed. While workable for small sets of rules, this approach would become unwieldy in large production systems. In the future it would be useful to find ways of more effectively choosing which rules to apply earlier on so as to avoid needless backtracking and to produce the better translations first. One solution suggested by the Verbmobil transfer algorithm (Buschbeck-Wolf, 1996) and the METAL system (Hutchins, 1992) would be to order the rule application by complexity, where complex rules are preferred to simple ones. Verbmobil does this automatically, while METAL has manually assigned

rankings. The ordering of two rules would be done through a comparison of the number and type of unification equations in both rules. Constraint equations would be considered most restrictive, with pseudo-unification with a value being the next most restrictive, and pseudo-unification between two paths being least restrictive.

Another possible solution would borrow from the Multi-engine MT system method of using language modeling to help choose among alternate translations (Frederking, 1997). This is also the approach taken by the Corelli system at NMSU (Zajac, 1999). A multi-engine MT system takes as input the translation output from various types of MT technologies, such as transfer, example-based, or even direct MT. The various inputs are put into a chart where translation segments are chosen based on a language model of the target language. The segments chosen, which maximize the probability of the translation according to the language model, are then concatenated together to produce the final translation. The AVENUE transfer engine can produce multiple translations which can all be fed into a multi-engine MT system. One possible drawback of using language modeling occurs when translating into the less-commonly spoken language. The language may not have many written materials and the language models produce for the language may not

be very rich.

To test the ability of the system to improve translation output, the Chinese to English grammar will be augmented with a variety of rules to handle translation of different types of noun phrases. The system has already been adapted to be able to participate in the Chinese to English multi-engine MT system mentioned in the Results chapter. The transfer rules' explicit structural reordering abilities should be able to help improve scores. The MEMT score on the test set of 993 Chinese sentences without the transfer engine would be compared with scores resulting from adding in the transfer engine.

6.2 Robustness

More work can be done to separate the analysis, transfer, and generation lexicons to make them more flexible and to reduce the time needed to do lexical transfer, as well as being able to develop the lexicons separately. With separate lexicons a source word could still contribute to the analysis, even if it could not be translated by the transfer lexicon. In these cases either the word could be left out of the translation or the original source word could be inserted.

6.3 Preprocessors

Some languages may have special processing needs that cannot be met with the above rule formalism. For example, languages such as Chinese or Japanese that do not mark word boundaries in text need to be segmented before translation could occur. Other possible preprocessing stages would be to identify times, dates, numbers, money amounts, and other entities so that they could be specially translated or translated as single units. To accommodate such possibilities, the system should have a flexible interface to add in the preprocessing modules specific to a particular language. The preprocessors would have a standardized interface to interact with the transfer engine so that they could easily be created and added.

6.4 Compounds

Compounds and idioms are frequently used in language, but the transfer engine can only work with single word lexical units. Adding in the ability to handle compounds and idioms is a near-term goal and will be necessary to use the system on most types of text.

6.5 Semantic/Discourse Analysis

Currently, AVENUE has no plans to try to learn deep semantic analysis or discourse processing for anaphora resolution. For the engine itself to be as widely useful as possible as a translation platform, it would be useful to add these as optional components. Such components would likely be a full research project in and of themselves, and adding them would be a long-term goal.

Appendix A

Sample Chinese to English Transfer Rules

```
;; -*- coding: utf-8 -*-
;; English-Chinese Transfer Rules

(topnode S)

; Transfer declarative sentences
{S,1}
S::S : [NP VP] -> [NP VP]
(
  (x1::y1)
  (x2::y2)
  ((x0 subj) == x1)
  ((x0 subj case) = nom)
  (x0 = x2)

  (y0 = x0)

  (y1 == (y0 subj))
  (y2 = y0)

  ((y0 agr) = (y2 agr))
  ((y1 agr) = (y2 agr))
)

; Transfer question sentences
{S,2}
S::S : [NP VP MA] -> [V NP VP "?"]
(
  (x1::y2)
  (x2::y3)
  ((x0 subj) = x1)
  ((x0 subj case) = nom)
  ((x0 act) = quest)
  (x0 = x2)

  (y0 = x0)

  ((y0 act) =c quest)

  ((y1 form) = do)
  ((y1 agr) = (y2 agr))
  (y2 == (y0 subj))
  (y3 = y0)
)

; Chinese doesn't use a
; definite article like "the"
; so insert one for common nouns
{NP,1}
NP::NP : [NB] -> ["the" NB]
(
  (x1::y2)
  (x0 = x1)

  (y0 = x0)

  (y2 = y0)
  ((y2 ntype) = common)

  ((y0 agr) = (y2 agr))
```

```
  ((y0 ntype) = (y2 ntype))
)

; Transfer n-bar structures
{NP,2}
NP::NP : [NB] -> [NB]
(
  (x1::y1)
  (x0 = x1)

  (y0 = x0)

  (y1 = y0)
  ((y0 agr) = (y1 agr))
  ((y0 ntype) = (y1 ntype))
)

; Transfers determiner phrases,
; deleting Chinese measure word
{NP,3}
NP::NP : [DET M NB] -> [DET NB]
(
  (x1::y1)
  (x3::y2)
  (x0 = x1)

  (y0 = x0)

  ((y2 case) = (y0 case))
  ((y2 num) = (y0 num))

  ((y0 agr) = (y2 agr))
  ((y0 ntype) = (y2 ntype))
)

; Transfers nouns
{NB,1}
NB::NB : [N] -> [N]
(
  (x1::y1)
  (x0 = x1)

  (y0 = x0)

  ((y1 case) = (y0 case))
  ((y1 num) = (y0 num))

  ((y0 agr) = (y1 agr))
  ((y0 ntype) = (y1 ntype))
)

; Transfers stative-verb modifiers
; to adjective noun
{NB,2}
NB::NB : [VS DE N] -> [ADJ N]
(
  (x1::y1)
  (x3::y2)
  (x0 = x3)
  ((x0 mod) = x1)

  (y0 = x0)

  ((y2 case) = (y0 case))
  ((y2 num) = (y0 num))

  ((y0 agr) = (y2 agr))
  ((y0 ntype) = (y2 ntype))
)


; Transfers transitive verb constructions
{VP,1}
VP::VP : [V NP] -> [V NP]
(
  (x1::y1)
  (x2::y2)
  ((x2 case) = acc)
  ((x0 obj) = x2)
  (x0 = x1)

  (y0 = x0)

  ((y2 case) = (y0 obj case))
;  (y1 = y0)

  ((y0 agr) = (y1 agr))
)

; Transfers intransitive verbs
```

```
{VP,2}
VP::VP : [V] -> [V]
(
  (x1::y1)
  (x0 = x1)

  (y0 = x0)

  ;(y1 = y0)
  ((y0 agr) = (y1 agr))
)

; Handles conversion of Chinese stative verb
; to the "be" ADJ structure English uses
{VP,3}
VP::VP : [VS] -> [V ADJ]
(
  (X1::Y2)
  (x0 = x1)

  (y0 = x0)

  ((y1 form) = be)

  ((y0 agr) = (y1 agr))
)
```

Bibliography

[1] Akiba, Yasuhiro, Megumi Ishii, Hussein Almuallim, Shigeo Kaneda. 1995. Learning English Verb Selection Rules from Hand-made Rules and Translation Examples. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*. Leuven, Belgium. July.

[2] Arnold, Doug and Louis des Tombe. 1987. Basic Theory and Methodology in EUROTRA. In Sergei Nirenburg, editor, *Machine Translation: Theoretical and Methodological Issues*. Cambridge University Press, Cambridge.

[3] AppTek. 2002. TranSphere. Available at http:// www.apptek.com/ AppTek/ Products/ MachineTranslation/ TranSpheremore_page.htm

[4] Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Blackwell Publishers, Inc. Masden, Massachusetts.

[5] Buschbeck-Wolf, Bianka and Christel Tschernitschek. 1996. What you Always Wanted to Know About Semantic Transfer. Technical Report, Institute for Logic and Linguistics, IBM Informationssysteme GmbH, September.

[6] Dorna, Michael and Martin Emele. 1996. Efficient Implementation of a Semantic-based Transfer Approach. Technical Report, Universität Stuttgart, July.

[7] Frederking, R., Rudnicky, A., and Hogan, C. 1997. Interactive Speech Translation in the DIPLOMAT Project. Presented at

the Spoken Language Translation workshop at the 35th Meeting of the Association for Computational Linguistics, ACL-97. Madrid, Spain.

[8] Han, Benjamin. 2001. UKernel: A Unification Kernel. Available at http://www-2.cs.cmu.edu/~ benhdj/c_n_s.html.

[9] Her, O.S., D. Higinbotham, and J. Pentheroudakis. 1991. An LFG-based machine translation system. *Computer Processing of Chinese and Oriental Languages*, 5(3):285-297, November.

[10] Hutchins, W. J. 1986. *Machine Translation: Past, Present, and Future*. Halsted Press, New York.

[11] Hutchins, W. John and Somers, Harold L. 1992. *An Introduction to Machine Translation*. Academic Press, London.

[12] Koskenniemi, Kimmo. 1983. Two-level morphology: A general computational model for word form recognition and production. Publication No. 11, Department of General Linguistics, University of Helsinki.

[13] Leech, G., R. Garside and M. Bryant. 1994. CLAWS4: The tagging of the British National Corpus. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 94)* Kyoto, Japan, pp. 622-628.

[14] Nirenburg, Sergei, et.al. 1992. *Machine Translation: A Knowledge-Based Approach*. Morgan Kaufmann Publishers, Inc. San Mateo, California.

[15] Nirenburg, Sergei and Raskin, Victor. 1998. Universal Grammar and Lexis for Quick Ramp-up of MT Systems. In *COLING-ACL '98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 975-979, Montreal, Quebec Canada.

[16] Papineni, Kishore, Salim Roukos, Todd Ward, Wei-Jing Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. IBM Research Report, IBM Research Division, Thomas J. Watson Research Center. September.

[17] Probst, Katharina, Ralf Brown, Jaime Carbonell, Alon Lavie, Lori Levin, and Erik Peterson. 2001. Design and Implementation of Controlled Elicitation for Machine Translation of Low-Density Languages. In *Workshop MT 2010 at Machine Translation Summit VIII.*

[18] Probst, Katharina and Lori Levin. 2002. Challenges in Automated Elicitation of a Controlled Bilingual Corpus. In TMI 2002.

[19] Probst, Katharina. Unpublished. "Semi-automatic Learning of Transfer Rules for Machine Translation of Low-Density Languages".

[20] Rosé, C. P. and A. Lavie. 2001. Balancing Robustness and Efficiency in Unification-augmented Context-Free Parsers for Large Practical Applications. In van Noord and Junqua, editors, *Robustness in Language and Speech Technology*, ELSNET. Kluwer Academic Press. March.

[21] Tomita, Masaru, Editor, Teruko Mitamura, Hiroyuki Musha, and Marion Kee. 1988. The Generalized LR Parser/Compiler Version 8.1: User's Guide. CMU Center for Machine Translation Technical Report. April.

[22] Trujillo, Arturo. 1999. *Translation Engines: Techniques for Machine Translation.* Springer-Verlag London Limited, London.

[23] Unicode Consortium. 1996. *The Unicode Standard, Version 2.0.* Addison-Wesley Developers Press, Reading, Massachusetts.

[24] Wu, Dekai and Hongsing Wong. 1998. Machine Translation with a Stochastic Grammatical Channel. In *COLING-ACL '98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 1408-1414, Montreal, Quebec Canada.

[25] Zajac, Rémi. 1999. A Multilevel Framework for Incremental Development of MT Systems. In *Proceedings of the Machine Translation Summit VII*, National University of Singapore, September 13-17, 1999, Singapore. pp.646-653.

[26] Zhang, Ying, Ralf D. Brown, and Robert E. Frederking. Adapting an Example-Based Translation System to Chinese. To appear in *Proceedings of Human Language Technology Conference 2001 (HLT-2001)*.