Translingual Grammar Induction for Conversational Systems

by

John Sie Yuen Lee

BMath in Computer Science
University of Waterloo, Canada (2002)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

Author .....
Department of Electrical Engineering and Computer Science
June 9, 2004

Certified by .....
Stephanie Seneff
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Translingual Grammar Induction for Conversational Systems

by

John Sie Yuen Lee

Submitted to the Department of Electrical Engineering and Computer Science
on June 9, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

We propose an induction algorithm to semi-automate grammar authoring in an interlingua-based machine translation framework. This algorithm is designed for restricted domains within the context of multilingual conversational systems. It uses a pre-existing one-way translation system from some other language to the target language as prior information. It then infers a grammar for the target language.

We demonstrate the system's effectiveness on a weather domain and on a travel domain. We automatically induced Chinese and French grammars for these domains from their English counterparts, and then showed that they can produce high-quality interlingua to be used in translation.

Thesis Supervisor: Stephanie Seneff
Title: Principal Research Scientist

Acknowledgments

I would like to thank my advisor Dr. Stephanie Seneff, for her generosity with her time and attention. She devoted many hours — far exceeding what a student could hope for — answering my questions, suggesting new ideas, and even debugging my code.

This research was in part supported by a fellowship from the National Sciences and Engineering Research Council of Canada, and by the NTT Corporation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For more than a decade, the Spoken Language Systems group has been conducting research leading to the development of conversational systems. These systems are capable of interaction in spoken dialogue in a variety of languages, including English [26], Japanese [21] and Chinese [24]. They enable naive users to access and manage information in a variety of domains, such as weather [26], travel [27], and task delegation [20].

For all domains and languages, a common framework for natural language understanding (NLU) and natural language generation (NLG) is adopted. In the NLU component, a language-independent meaning representation, or *semantic frame*, is extracted from the user input. This representation facilitates effective communication with the application back-end, the dialogue management and the discourse context resolution components.

Within this framework, a language learning system has recently been introduced [19]. A native speaker of Chinese who wishes to learn English, for example, can speak a sentence in his/her native tongue and have the system paraphrase it in English. He/she can then attempt to repeat the English sentence to advance a dialogue with the system in English. Two criteria that would be useful evaluation metrics for such research are:

1. The effectiveness of the framework for translation, at least in restricted domains.

2. The ease of writing grammars that extract a meaning representation from user input in multiple languages.

This thesis focuses on improving performance in the second criterion. Currently, grammar authoring is a laborious, error-prone process that demands a lot of expertise and pa-

tience. In many domains of interest to us, we already have mature, high-quality grammars in place for at least one language, L (in most cases, English). We propose here a grammar induction algorithm that leverages such pre-existing grammars to semi-automate grammar authoring in another language L'.

We evaluate our induction approach by comparing induced grammars and hand-crafted grammars with respect to the first criterion. Our experiments in two restricted domains demonstrate that the induced grammar can generate high-quality translation from L' back to L.

The intent of this document is to provide the reader with an understanding of our induction approach. The chapters are divided as follows:

- **Chapter 2: Related Work**
  in which we review previous work on grammar induction.
- **Chapter 3: Framework Overview**
  in which we outline the NLU and NLG frameworks within which our grammar induction algorithm functions.
- **Chapter 4: Word Alignment**
  in which we describe the L-L' word alignment process.
- **Chapter 5: Tree Transformation**
  in which we illustrate the transformation of an L parse tree into its L' counterpart.
- **Chapter 6: Trace Simulation**
  in which we infer constituent movements in the L' parse tree.
- **Chapter 7: Evaluation**
  in which we report experiments and results for grammar induction in two different domains and target languages.
- **Chapter 8: Future Work**
  in which we suggest possible enhancements for grammar induction.
- **Appendix A: Implementation Details**
  in which we provide details of the implementation of the induction algorithm.

Chapter 2

Grammar Induction

Grammar induction is the process of inferring the structure of a language, given a corpus of sentences drawn from it. The structure is typically expressed as parse trees. The grammar, then, evaluates the probability of trees.

The techniques used in the induction and the nature of the induced grammar vary considerably. Below are some of the main factors for these variations:

- **Prior information**: This could range from no information to a treebank of parsed sentences. Sometimes there might be examples from related domains or languages. In general, a broad domain, such as the Wall Street Journal [13], would require more prior information than a narrow one, such as the Air Travel Information Service domain [17].

- **Purpose of the grammar**: The grammar might be used to define a *parsing model*, i.e., to propose one or more parse trees for a sentence with associated probabilities. Typically, the parse tree is post-processed to extract some further information for specific natural language processing tasks. For example, predicate-arguments might be extracted from a syntactic parse tree, or an interlingua from a semantic parse tree. In this case, the parse tree, and hence the grammar, must provide meaningful labels. Normally, some prior information, or manual post-editing of the induced grammar, is necessary.

  The grammar might also be used primarily as a *language model*, i.e., to give the probability of a sentence. For example, in speech recognition, such a model would help constrain the search space. In this case, the objective of the grammar is not so

much to give meaningful parses, but rather to capture patterns or structures in the sentences, in order to maximize the likelihood of sentences. Often there is minimal or no prior information available on sentence structure.

2.1 Grammar Induction for Language Models

N-grams, which may be considered the most basic grammar, could be directly estimated from the corpus. For more sophisticated types of grammar, a common strategy is the following: Assume a type of grammar (e.g., context-free grammar); generate a naive, initial grammar from the corpus; define some function that measures how well the grammar fits the corpus, then incrementally modify the grammar (e.g., by adding or removing a rule) to optimize that function. Examples of such a function include a divergence measure and mutual information [14, 22]; in other systems the function is optimized via the Inside-Outside Algorithm [16] or in a Bayesian framework [4].

2.2 Grammar Induction for Parsing Models

The work described in this thesis falls into this category. The induction approach depends largely on the prior information available.

2.2.1 Induction from In-domain Examples

For a broad domain, such as news, the dominant approach is to use a treebank of parsed example sentences as training corpus. The corpus is used to estimate parameters in a generative parsing model. Charniak argued in [2] that one could construct a high-quality parser simply by reading the parent-child relationships in the trees as grammar rules. Both Collins [6] and Charniak [3] later did without an explicit grammar.

This approach is often infeasible for smaller domains, since large corpora of parsed sentences are less likely to exist.

2.2.2 Induction from Other domains

In this approach, an existing grammar in a related domain is used as a starting point. Hwa [9] investigated the adaptation of a grammar from one domain to another (e.g., from the

Wall Street Journal to the ATIS corpus). She found that, even with a corpus that has only high-level constituent information for the new domain, the adapted grammar consistently outperforms a directly induced grammar.

Wang and Acero [25] make use of a 'grammar library', which contains common, cross-domain concepts as part of the prior information for grammar induction. They also exploit syntactic constraints in English, which are applicable for multiple domains. Similarly, Lavie et al. [11] ported semantic grammars from one domain to another.

2.2.3 Induction through User Interaction

In Gavaldà and Waibel's GSG system [8], prior information is supplied by the user in the form of a 'Domain Model', from which a basic context-free grammar is automatically generated. When the system encounters a sentence that it could not parse, an interactive learning episode starts. During the episode, the system solicits information from the user in a natural-language dialogue, and constructs a parse tree for the sentences. It integrates new rules into the grammar, and updates its Prediction Models using the hypothesized parse trees. It also edits the rules and detects subsumptions and ambiguities.

The main strength of GSG is its ability to grow and refine the grammar through learning episodes. The developer is thus able to incrementally expand the coverage of the grammar.

2.2.4 Induction from Other Languages

In [23], which is most closely related to our work, the prior information is a grammar for some language L. A native speaker of L' provides pairs of aligned sentences in L and L'. The induction algorithm transforms the L parse trees into L' parse trees. With no knowledge of the structure of the L' language beyond the word alignments, the algorithm is sometimes forced to make rather arbitrary assumptions, especially when re-ordering branches and inserting new ones. The algorithm was used to induce a Polish grammar from an English grammar in a domain for physical symptoms. On a test set of 39 sentences, the induced grammar achieved 52% coverage of key-value pairs in the meaning representation.

Our approach is in fact quite similar to hers, with the following main difference. We assume the existence of an L' generation grammar as part of the translation capability from L to L'. Instead of asking the user to provide word alignments between L and L' sentences, we infer the alignments. The L' generation grammar also provides information about word

dependencies in L', which is useful in hypothesizing new trees in L'.

Chapter 3

Framework Overview

In an interlingua-based translation framework, two functions need to be implemented for each language L:

- PARSE(S_L, P_L) $\longrightarrow F$, which maps S_L, a sentence in the language L, to a language-dependent meaning representation F, according to some grammar P_L; and
- GEN(F, $G_{L'}$) $\longrightarrow S_{L'}$, which maps a meaning representation F to a surface string in the language L', according to some grammar $G_{L'}$.

The translation of S_L in the language L' is, thus, $S_{L'}$ = GEN(PARSE(S_L, P_L), $G_{L'}$).

In our case, the function PARSE is performed by the natural language understanding (NLU) system TINA [18]; the function GEN is performed by the natural language generation (NLG) system GENESIS [1]; the meaning representation is a *semantic frame*. In the next two sections we give an overview of these systems and representations.

3.1 Meaning Representation

Our design of the meaning representation is inspired by Government-Binding theory [5], which posits that a sentence has both a *surface form*, and a *deep structure* that represents its underlying meaning. While the deep structure is shared across different languages, the surface form is language-dependent, and is derived from the deep structure by constituent movements called in [10] the "move-α" paradigm.

In our conversational systems, the meaning representation is a semantic frame, a language-independent, hierarchical structured object that encodes meaning. It is a frame that con-

```
{clause verify
        :auxil "will"
        :topic {topic pronoun
                      :name "it" }
        :pred {pred rain
                    :pred {pred temporal
                                :topic {topic weekday
                                              :name "tomorrow" } } } }
```

Figure 3-1: Semantic frame for the utterance *Will it rain tomorrow?* { ... } designates a frame, which in our framework may be one of three major classes: a clause, a topic or a predicate.

tains key-value pairs. The values could, in turn, be frames, or simple strings, written in English. Figure 3-1 shows a semantic frame representation of the utterance *Will it rain tomorrow?*.

3.2 Natural Language Understanding

TINA decomposes the PARSE function into two steps, FRAME $\circ$ TREE:

1. TREE(S_L, P_L.`rules`, P_L.`constraints`), which maps S_L to a parse tree T_L according to:

   - P_L.`rules`, a set of probabilistic context-free rules.
   - P_L.`constraints`, a set of semantic constraints for constituent movements.

2. FRAME(T_L, P_L.`actions`, P_L.`translations`), which maps T_L into a semantic frame according to:

   - P.`actions`, a set of instructions that guide designated nodes in T_L to create frames or assign key values in the frame under construction[1].
   - P_L.`translations`, a lexicon that translates key values in the semantic frame from L into English.

A TINA grammar P_L thus consists of four parts: P_L.`rules`, P_L.`constraints`, P_L.`actions`, and P_L.`translations`. We now examine the role of each of these parts.

[1]In practice, P.`actions` often differs slightly from one language to another. Our grammar is induced such that it would work with the P.`actions` for English.

`sentence`	$\longrightarrow$	`full_parse`
`full_parse`	$\longrightarrow$	`question`
`question`	$\longrightarrow$	`do_question`
`do_question`	$\longrightarrow$	`auxil subject predicate`
...		
`weekday`	$\longrightarrow$	*tomorrow*
`weather_verb`	$\longrightarrow$	*rain*
...		

Table 3.1: Some rewrite rules taken from $P_{English}$.`rules` for JUPITER, a weather domain.

3.2.1 Parsing Rules

As in most NLU systems, P_L.`rules` is a set of context-free rules that describe the sentence structure in the language L. The grammars we are working with incorporate both syntactic and semantic information simultaneously.

At the higher levels of the parse tree, major syntactic constituents, such as `subject`, `predicate`, `object`, etc., are explicitly represented through syntax-oriented grammar rules. The syntactic structures tend to be domain-independent, capturing general syntactic constraints of the language. The upper half of Table 3.1 shows some of these rules for our English weather domain, JUPITER.

Near the leaves of the parse tree, major semantic classes, such as `weather_verb`, `date_name`, etc., are constructed according to semantic-oriented grammar rules. The semantic structures tend to be domain-dependent, capturing specific meaning interpretations in a particular application domain. The lower half of Table 3.1 shows some of these rules for JUPITER. The rules in Table 3.1 are a subset of those needed to parse the utterance *Will it rain tomorrow* to the tree in Figure 3-2.

3.2.2 Movement Constraints

Up to this point, our parse tree reflects the surface form of the sentence. Due to differences in word order and structure between the languages L and L', two equivalent sentences, S_L and $S_{L'}$ may have parse trees with substantially different hierarchical structures. However, in theory, they share the same deep structure, and PARSE should map them to identical semantic frames. To facilitate the construction of these frames, TINA uses a trace mechanism to move tree branches, so that their final positions would resemble the deep structure, or

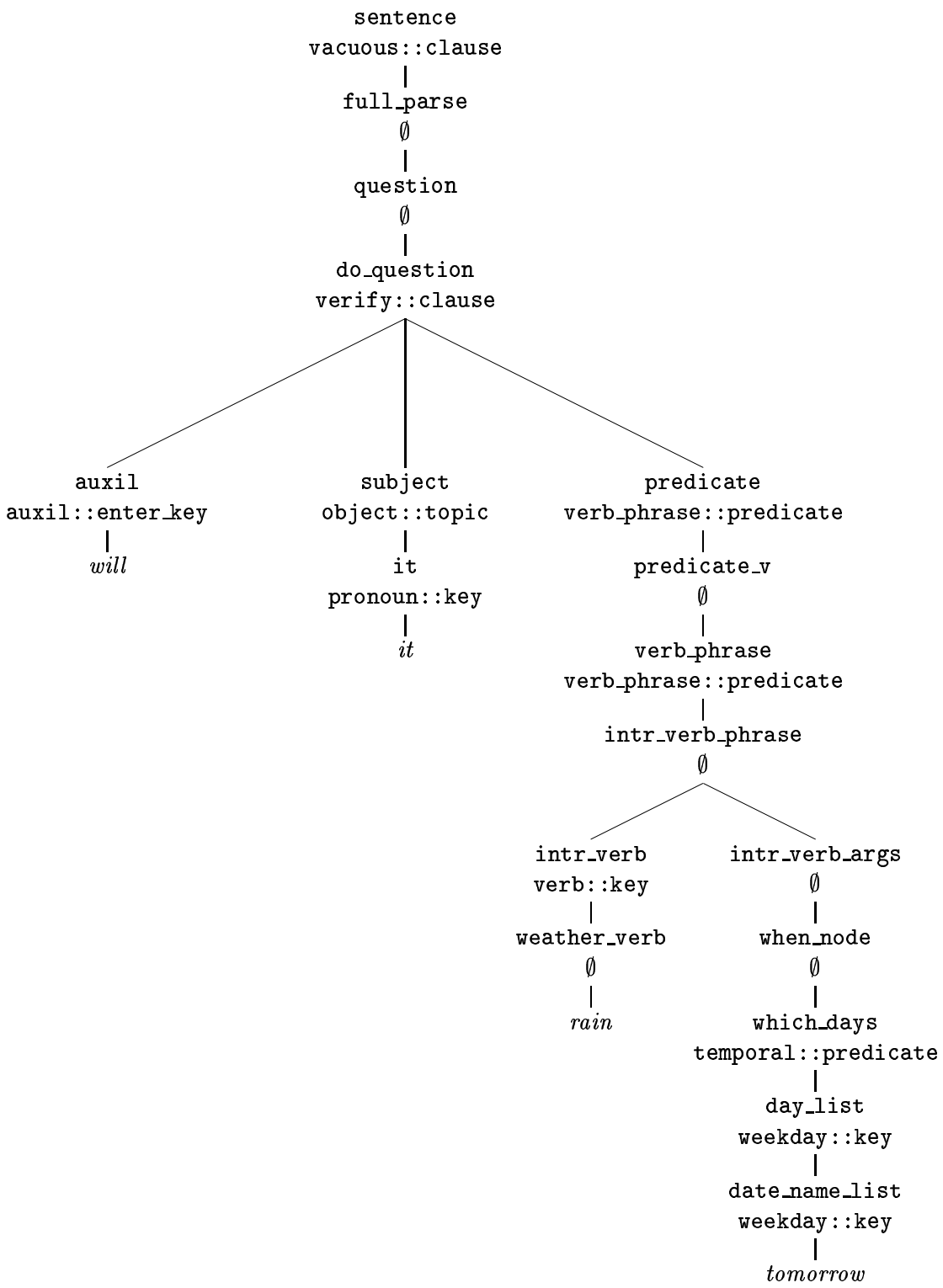


Figure 3-2: Parse tree for *Will it rain tomorrow.* The top label in a node is given by $P_{English}$.`rules` (see §3.2.1). The bottom label, given by P.`actions` (see §3.2.3), is of the form <`name`>::<`type`>, specifying the type of component to be created in the frame, and the name of that component.

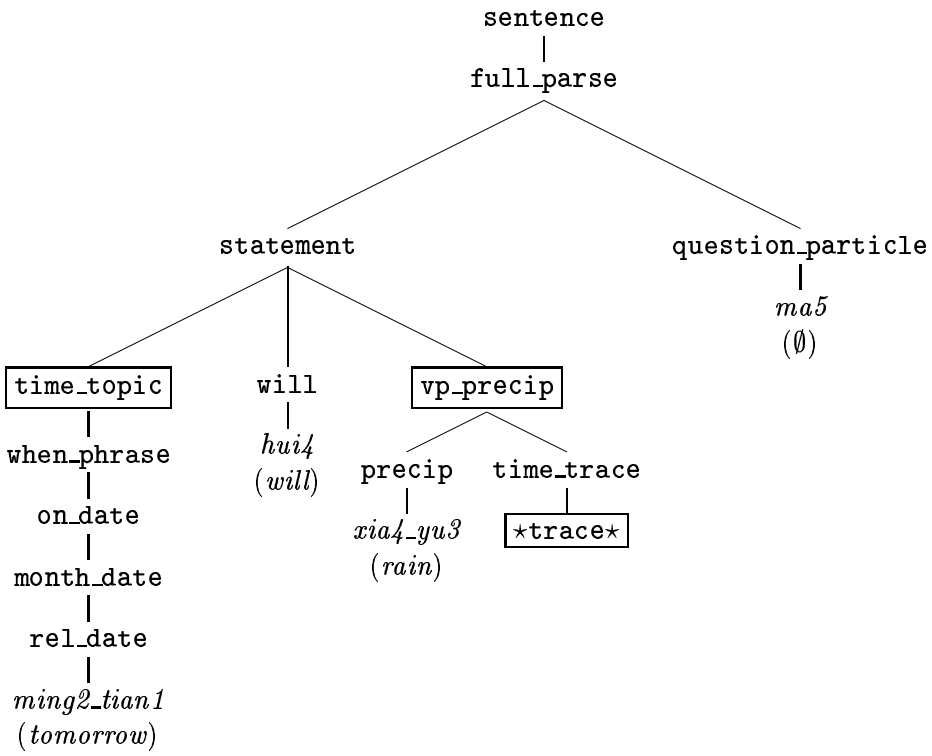


Figure 3-3: Parse tree for *ming2_tian1 hui4 xia4_yu3 ma5*. The trace is licensed through a process involving a designated generator `time_topic`, an activator `vp_precip`, and a trace tag ⋆`trace`⋆.

the canonical hierarchy in the semantic frame.

According to Government-Binding theory, the object of a verb or a preposition may be moved during surface form realization, leaving behind a *trace* in its original position. An example is the sentence *(What)*$_1$ *did she say* (t_1)*?*, where *what* is moved from its original position, t_1, to the front of the sentence.

Many other languages exhibit constituent movements that could be handled elegantly as traces for our computational purposes, even though they are not traces as defined in [5]. An example is *topicalization* in Chinese [12], where temporal and locative phrases could be moved forward. For example, *Will it rain tomorrow* is translated in Chinese as *(ming2_tian1)*$_1$ *hui4 xia4_yu3* (t_1) *ma5*, whose topic, the temporal phrase *ming2_tian1* (*tomorrow*), is moved to the front of the sentence.

TINA uses a common mechanism to handle all constituent movements[2]. An important goal of this research is to both detect when a trace is needed and to provide the developer

[2]In the rest of the thesis, we will use the term *trace* broadly to denote these constituent movements.

with a specification of its encoding. In this thesis, a trace is deemed necessary when the word order and hierarchy in S_L and $S_{L'}$ are so different that it is impossible to transform $T_{L'}$ to the original T_L simply by changing the order of siblings. In other words, the hierarchical structure of the parse tree of $S_{L'}$ must be altered. For example, in the parse tree for *ming2 tian1 hui4 xia4 yu3 ma5* in Figure 6-2, the branch for *tomorrow* is a sibling of *will*, whereas in Figure 3-2 it is a sibling of *rain*. No rearrangement of the siblings could transform this Chinese parse tree to its English counterpart.

The trace mechanism is controlled by P_L.`constraints`, which defines the nodes from which, and to which, branches may be moved, as well as the semantic constraints for these movements. Formally, it contains:

- A set of *semantic categories*, e.g., { `time, location, complement ...` }.
- A set of semantic category assignments for selected parse nodes, e.g., `time_topic` belongs to the category `time`, `location_topic` belongs to `location`, and `complement_phrase` belongs to `complement`.
- A set of *generators*, e.g., `time_topic`, `location_topic`, and `complement_phrase`. These are nodes whose subtree may be moved to a different location in the parse tree.
- A set of *activators*, e.g., `vp_precip`, `hotel_thing`, and `vp_take_time`. These are nodes whose descendants are obliged to absorb a generated constituent. Each activator node is restricted to activate only those generated constituents that match its assigned *activation categories*, e.g., `vp_precip` is assigned {`location, time`}, `hotel_thing` is assigned {`location`}, and `vp_take_time` is assigned {`complement`}.
- A set of *absorbers*, denoting the original location of the moved constituent.

Every subtree whose root node is a generator is considered a candidate for trace movement, or, in short, *generated*. The candidate is tagged with the union of the semantic categories of its descendant nodes. The candidate then searches for a valid activator among its siblings to the right. A valid activator must have, as one of its activation categories, one of the semantic categories tagged to the candidate. If such an activator is found, the generated subtree is grafted onto the first appropriate trace node among the activator's descendants.

Node		Instruction
predicate	$\longrightarrow$	verb_phrase
which_days	$\longrightarrow$	temporal
...		
vacuous, verify	$\longrightarrow$	clause
pronoun, weekday	$\longrightarrow$	topic
verb_phrase, temporal	$\longrightarrow$	predicate
verb, auxil	$\longrightarrow$	key
weekday, pronoun	$\longrightarrow$	named_topics

Table 3.2: Some rewrite rules from P.actions for JUPITER.

In our example in Figure 6-2, the subtree whose root is time_topic is a candidate for trace with the semantic category time. Since the node vp_precip is an activator that has the same activation category, the subtree under time_topic is moved under the node trace. This yields a Chinese parse tree that is identical in structure to its English counterpart in Figure 3-2.

Trace movements complete the TREE function and output a tree T_L that corresponds to the 'deep structure' of S_L and is well-positioned to construct a semantic frame.

In the induction process, we will use T_L as a basis to hypothesize an equivalent parse tree $T_{L'}$.

3.2.3 Actions

Since the trace mechanism produces parse trees that correspond to the deep structure, TINA could derive the semantic frame directly from the tree. It does so by assigning instructions to designated nodes to guide the construction of the frame. The bottom label of the nodes in Fig 3-2 shows these instructions.

P.actions assigns these instructions to nodes in T_L. A sample from P.actions, for the weather domain, JUPITER, is shown in Table 3.2. There are two main types of instructions. Those in the upper half of the table give the semantic names of the node; those in the lower half specify whether the node is to contribute a clause, a predicate, a topic, or a key to the frame.

In the induction process, we will track this frame construction process to create an alignment between words in S_L and the components in the semantic frame.

Key	$w_{Chinese}$	$w_{English}$
`date`	*ming2_tian1*	*tomorrow*
`date`	*xing1_qi1_yi1*	*monday*
`date`	*xing1_qi1_er4*	*tuesday*
`quality`	*leng3*	*cold*
`quality`	*re4*	*hot*

Table 3.3: Some entries from $P_{Chinese}$.`translations` for JUPITER.

`verify`	$\longrightarrow$	($if :rhet >exist >verify1) :q_particle
verify1	$\longrightarrow$	($if :complement >wh_complement >verify2)
verify2	$\longrightarrow$	:topic ($if :phatic_pronoun >pull_time_loc) >auxil >preds
pull_time_loc	$\longrightarrow$	<==`temporal`
`temporal`	$\longrightarrow$	:topic
topic_template	$\longrightarrow$	(:name $core)
preds	$\longrightarrow$	... `rain` ...
`predicate_template`	$\longrightarrow$	(:verb $core)

Table 3.4: A subset of $G_{Chinese}$.`order` used in generating *ming2_tian1 hui4 xia4_yu3 ma5*.

3.2.4 Key Value Translation

English is the designated language for the key values in the semantic frame. When creating a frame from a non-English language, the key values are translated into English. $P_{L'}$.`translations` is a list of three-tuples in the form of ($w_{English}$, $w_{L'}$, *component*). This tuple means that whenever the word $w_{L'}$ is a key value under the frame or key called *component*, it is to be translated to $w_{English}$. A portion of $P_{Chinese}$.`translations`, taken from the Chinese weather domain, MUXING, is shown in Table 3.3.

In the induction process, we will automatically infer $P_{L'}$.`translations` from the L-interlingua-L' alignment.

3.3 Natural Language Generation

GENESIS decomposes the GEN function into two steps, VERBALIZE $\circ$ PREPROCESS:

1. PREPROCESS(F, $G_{L'}$.`pre`) $\longrightarrow$ $F_{L'}$, which maps a language-independent semantic frame F to a frame, $F_{L'}$, that is tailored for surface string realization in the language L'.

sister	
$w_{Chinese}$	Feature
jie3_mei4	$\emptyset$
jie3_jie5	$:older
mei4_mei5	$:younger

Table 3.5: Entry for *sister* in $G_{Chinese}$.lexicon.

- $G_{L'}$.pre [7] is a set of grammar rules that edit key values in F with the goal of facilitating language generation in L'. Typically, the key values concerned encode linguistic features. For example, a $G_{Spanish}$.pre might add gender information for nouns.

2. VERBALIZE($F_{L'}$, $G_{L'}$.order, $G_{L'}$.lexicon) $\longrightarrow$ $S_{L'}$, which paraphrases a pre-processed semantic frame $F_{L'}$ in language L' according to:

   - $G_{L'}$.order, a set of rewrite rules that specify the order in which the components of $F_{L'}$ are to be processed.
   - $G_{L'}$.lexicon, a lexicon that maps key values in the frame components to surface strings in L'.

3.3.1 Feature Propagation

Hardly any component in a semantic frame could be paraphrased in isolation. Word-sense disambiguation and inflectional endings, to name just two decisions among many other, need to be resolved by syntactic, semantic and prosodic contexts from other components of the frame.

GENESIS provides these contexts in two ways. In PREPROCESS, it inserts the relevant context by explicitly adding or modifying key values. In VERBALIZE, it allows components to set *features* which are propagated along the generation process.

3.3.2 Lexicon

One component in our NLG framework that relies on features is the lexicon. For example, there are three ways to translate the word *sister* in Chinese, depending on her age with respect to the speaker. The lexicon entry in Table 3.5 lists these alternatives.

```
{q relationship
   :name "sister"
   :pred {p relative_age
            :topic "younger" } }
```

Figure 3-4: Semantic frame for *younger sister*.

In Figure 3-4, *younger sister* is represented as a `relationship` topic frame and a `relative_age` predicate. When paraphrasing in Chinese, the `relative_age` predicate does not generate any surface string, but simply sets the feature `$:younger`. When the `:name` key under `relationship` is processed, GENESIS looks up the lexicon for the *sister* entry. In this case, the feature determines that *mei4_mei5* is to be generated.

3.3.3 Generation Rules

Table 3.4 shows a portion of $G_{Chinese}$.`order` taken from JUPITER. There are two main types of rules. In the first type, the left hand side (LHS) is a frame name and the right hand side is the generation instruction. For example, the first rule in Table 3.4 says that the `verify` frame is to generate a question article, if the key `:q_article` exists.

In the second type, the LHS is simply a step name. The generation instruction found in the RHS is to be carried out for the frame that is being processed. For example, for the `verify` frame, we execute a sequence of steps – `verify1`, `verify2`, and `pull_time_loc` – while remaining in the same frame. Then, `pull_time_loc` takes us down to the `temporal` frame, for which there is a rule of the first type.

These rules clearly define the words generated by each frame component.

In the induction process, we will track the generation process to infer the sentence structure for $S_{L'}$, and to infer an alignment between words in $S_{L'}$ and the components in the semantic frame.

3.4 Induction Approach

We propose an induction algorithm that automatically infers $P_{L'}$. It takes as input the following three pieces of prior information:

- TRAIN$_L$: Training sentences in some other language L.

- P_L: The parse grammar for L.
- $G_{L'}$: The generation grammar for L'.

In theory, the development effort needed for adding a new language L' to the conversational system is then reduced to $G_{L'}$.

3.4.1 Algorithm Outline

We summarize below the major steps of the induction algorithm. For each sentence $S_L \in$ TRAIN$_L$:

1. **Translate S_L to $S_{L'}$**: Using P_L and $G_{L'}$,
   (a) Compute the parse tree T_L = TREE(S_L, P_L.`rules`, P_L.`constraints`).
   (b) Map T_L to the semantic frame F = FRAME(T_L, P.`actions`, P_L.`translations`).
   (c) Paraphrase F in L', $S_{L'}$ = GEN(F, $G_{L'}$).
2. **Infer L-L' word alignments**: During step 1b, infer an alignment between the words in S_L and the frame components in F (see §4.1). During step 1c, infer an alignment between the frame components and the words in $S_{L'}$ (see §4.2). Combining these two alignments yields an L-L' word alignment.
3. **Infer the sentence structure of $S_{L'}$**: During step 1c, infer a bracketing of the L' sentence (see §5.4.1).
4. **Construct $T_{L'}$**: Apply the following operations on T_L (obtained in step 1a) to transform it into $T_{L'}$:
   - Translate a leaf, using the L-L' word alignments (see Chapter 4).
   - Reposition a branch: Move a subtree in T_L to a new position to reflect the word order in L'. If the repositioned branch leaves behind a trace, insert appropriate trace nodes at its original and new positions (see Chapter 6).
   - Insert a branch: Insert new subtree(s) for each zero-to-one, zero-to-many, or one-to-many L-L' word alignment (see §5.4). Use the $S_{L'}$ bracketing to determine the point of insertion.

- Prune a branch: Prune the appropriate subtree(s) for each one-to-zero, many-to-zero or many-to-one word alignment (see §5.2).

5. **Grow $P_{L'}$ grammar**: Add to the following parts of the grammar:

   - $P_{L'}$.`rules`: Add rewrite rules read off from $T_{L'}$. Flag the new rules if they introduce left recursions.
   - $P_{L'}$.`translations`: Add entries to the lexicon based on the L-L' word alignment.

   The induced grammar is designed to use the same values for P.`actions` as were assigned for the English grammar. For trace movements, we induce a P.`constraints` which simulates a trace mechanism with no semantic constraints (see §6.1).

Chapter 4

Word Alignment

The first step in our induction algorithm is to align words in S_L, a sentence in the language L, to those in $S_{L'}$, a paraphrase of S_L in the language L'. In our interlingua-based translation framework, the semantic frame serves as the common link between S_L and $S_{L'}$. We split the alignment task into two subtasks:

1. During PARSE, we align L words to components in the semantic frame;

2. During GEN, we align components in the frame to L' words.

We then combine the two alignments to create an L-L' alignment.

In the rest of the thesis, we will use S_L = *Will it rain tomorrow* and $S_{L'}$ = *ming2_tian1 hui4 xia4_yu3 ma5* as our examples.

4.1 L-interlingua Alignment

We define the *yield* of a node in a parse tree as the words (i.e., the leaf nodes) that are descendants of that node. For example, the yield of the node `verb_phrase` in Figure 4-1 is *rain tomorrow*.

As described in §3.2.3, each component in a semantic frame is created by a node in the parse tree. In our alignment process, a leaf is aligned to the component of the frame that is created by the node closest to the leaf.

In practice, the algorithm proceeds as follows. When a node creates a frame, we align its yield to that frame. Suppose the yield is $w_1 w_2 \ldots w_5$. As we descend down the tree,

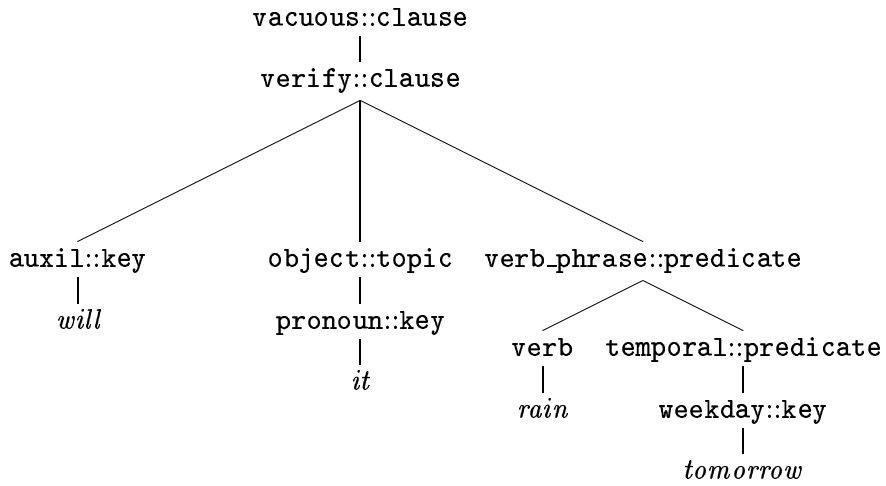


Figure 4-1: Parse tree for S_L, showing only the semantic labels given by P.`actions`.

another node might create a subframe. If the yield of this node is $w_1 w_2$, then we would narrow the alignment of the previous node to $w_3 w_4 w_5$.

We illustrate this process with an example. The parse tree for S_L is reproduced in Figure 4-1, but showing only the node labels that are used by P.`actions` to construct the semantic frame. We now work through the process of creating the semantic frame in Fig. 3-1:

1. The `vacuous` node creates a clause with the same name. The whole sentence is aligned to the clause.

   {c `vacuous` L=*will it rain tomorrow* }

2. The `verify` node renames the clause after itself. The alignment is unchanged.

3. The `auxil` node creates a key under the clause, and assign its yield, *will*, as its value. Since `auxil` is closer to the leaf than `verify`, we associate *will* to the `:auxil` key, and take *will* away from the alignment at the `verify` clause.

   {c `verify` L=~~*will*~~ *it rain tomorrow*
   `:auxil` "*will*" L=*will* }

4. The `object` node then creates an empty topic frame, which is aligned with its yield, *it*. As in the previous step, the word *it* is deleted from the alignment at the `verify` clause.

{c verify *L=~~will it~~ rain tomorrow*
:auxil "*will*" *L=will*
:topic {q null *L=*"*it*" } }

5. The child of object, pronoun, creates a key :name and assigns a value to the key:

{c verify *L=~~will it~~ rain tomorrow*
:auxil "*will*" *L=will*
:topic {q pronoun *L=~~it~~*
:name "*it*" *L=it* } }

6. Next, verb_phrase creates a predicate:

{c verify *L=~~will it rain tomorrow~~*
:auxil "*will*" *L=will*
:topic {q pronoun *L=~~it~~*
:name "*it*" *L=it* }
:pred {p verb_phrase *L=rain tomorrow* }

7. Then verb renames the predicate to rain.

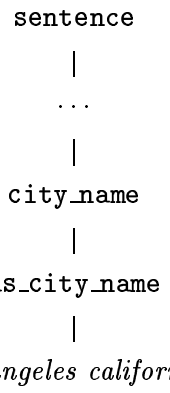
8. Lastly, temporal creates a predicate and a key under the predicate, assigning its yield as the key value.

{c verify *L=~~will it rain tomorrow~~*
:auxil "*will*" *L=will*
:topic {q pronoun *L=~~it~~*
:name "*it*" *L=it* }
:pred {p rain *L=rain ~~tomorrow~~*
:pred {p temporal *L=~~tomorrow~~*
:topic {q weekday
:name "*tomorrow*" *L=tomorrow* } } }

By now we have filled in all the L entries in the alignment in Figure 4-2.

4.1.1 Limitations

In some sentences, certain components of the semantic frame are not directly constructed by nodes in the parse tree. One example is the `in` and `state` frames for American states, which are produced upon separation of city and state names[1], independent of the parse tree. The parse tree of the utterance *Los Angeles, California* is:



and is represented as a `:topic` in the semantic frame:

`:topic` {q `city`
:name "*los angeles california*" }

Without taking any information from the parse tree, the `:topic` is further broken down into:

`:topic` {q `city`
:name "*los angeles*"
:pred {p `in`
:topic {q `state`
:name "*california*" } } }

Our current implementation does not track these further analyses, and would consider all three words, city and state, to be aligned to the `:name` key under the `city` frame.

[1] We turned off the function `pick_apart` when inducing a new grammar.

4.2 Interlingua-L' Alignment

To infer an interlingua-L' alignment, we observe the L' words generated by each component of the frame when it is verbalized as an L' string. As an example, we work through this process for the semantic frame in Figure 3-1, according to the Chinese generation grammar in §3.4. For the sake of clarity, we present these steps breadth-first rather than depth-first, which is the way it was implemented.

1. The rule

   `verify` $\longrightarrow$ ($if :rhet >exist >verify1) **:q_particle**

   generates the question particle *ma5* at the `verify` clause.

   {c `verify` L=$\emptyset$, L'=*ma5*
   `:auxil` "*will*" L=*will*
   `:topic` {q `pronoun` L=$\emptyset$
   :name "*it*" L=*it* }
   `:pred` {p `rain` L=*rain*
   :pred {p `temporal` L=$\emptyset$
   :topic {q `weekday`
   :name "*tomorrow*" L=*tomorrow* } } }

2. The rule

   verify2 $\longrightarrow$ ($if :phatic_pronoun >pull_time_loc) >**auxil** >preds

   verbalizes the `:auxil` key under the `verify` clause. After consulting the lexicon, *will* is realized as *hui4*.

   {c `verify` L=$\emptyset$, L'=*ma5*
   `:auxil` "*will*" L=*will* L'=*hui4*
   `:topic` {q `pronoun`
   :name "*it*" L=*it* L'=$\emptyset$ }
   `:pred` {p `rain` L=*rain*
   :pred {p `temporal`

:topic {q weekday

:name "*tomorrow*" *L=tomorrow* } } }

3. The pull_time_loc step executes a "pull" of the `temporal` predicates (<==`temporal`). The `temporal` predicate then generates, according to predicate_template, the `:name` key in the `weekday` topic:

   `topic_template` ⟶ (**:name** $core)

   which is mapped to the surface string *ming2 tian1*.

{c `verify` *L=∅, L′=ma5*
`:auxil` "*will*" *L=will L′=hui4*
`:topic` {q `pronoun`
`:name` "*it*" *L=it L′=∅* }
`:pred` {p `rain` *L=rain*
`:pred` {p `temporal`
`:topic` {q `weekday`
`:name` "*tomorrow*" *L=tomorrow L′=ming2_tian1* } } }

4. The >pred command processes the `rain` predicate, which is verbalized as *xia4_yu3* under the `predicate_template` step.

   predicate_template ⟶ (:verb $core)

{c `verify` *L=∅, L′=ma5*
`:auxil` "*will*" *L=will L′=hui4*
`:topic` {q `pronoun`
`:name` "it" *L=it L′=∅* }
`:pred` {p `rain` *L=rain* ~~*tomorrow*~~ *L′=xia4_yu3*
`:pred` {p `temporal` *L=*~~*tomorrow*~~
`:topic` {q `weekday`
`:name` "*tomorrow*" *L=tomorrow L′=ming2_tian1* } }
}

```
{ verify [L=null, L'=ma5]
  :auxil [L=will, L'=hui4]
  :topic { { q pronoun
           :name [L=it, L'=null] }
  :pred {p rain [L=rain, L'=xia4 yu3]
           :pred {p temporal
                    :topic {q weekday
                              :name [L=tomorrow,
                                     L'=ming2 tian1] } } } }
```

Figure 4-2: L-L' word alignment for *Will it rain tomorrow?*

The final alignment is shown in Figure 4-2.

4.2.1 Features

Multiple components in the semantic frame may propagate features to influence the generation of one L' word. For example, in Table 3.5, the generation of *mei4_mei5* depends on the feature $:younger, which is propagated by the relative_age predicate in Figure 3-4.

Since the word *mei4_mei5* is generated at the :name key, it is aligned to the word *sister*. However, it should be aligned to both *younger* and *sister*. We achieve this by remembering the features propagated by each component of the frame, and the features that are used by each component during lexicon look-up. If a component c_1 is aligned to $\emptyset$, but issues a feature that is later used in the generation of another component c_2, then the yield of the c_1 is also aligned to c_2.

4.2.2 Limitations

Just as the trace mechanism reconfigures the parse tree to enforce a canonical structure in the semantic frame, the PREPROCESS stage (see §3.3) manipulates the semantic frame to facilitate generation of surface strings in certain languages. For example, the utterance *I do not know of any bank machine in the vicinity* is represented by the semantic frame:

```
{c statement
   :topic {q pronoun
            :name "i" }
   :aux "do"
```

:negate "*not*"
:pred {p know
... } }

Before generation in Chinese, however, the pre-processor moves the :negate key inside the know predicate:

{c statement
:topic {q pronoun
:name "*i*" }
:aux "*do*"
:pred {p know
:negate "*not*"
... } }

Our current implementation does not keep track of these manipulations by the pre-processor.

4.3 Interlingua-L Alignment

It is possible to enhance our L-L' alignment by aligning the components of the frame to the words in S_L. Due to limitations such as the one discussed in §4.1.1, we might have an incomplete alignment. In particular, in the example in §4.1.1, *luo4_shan1_ji1* (Chinese for *Los Angeles*) would be aligned with *los angeles california*, while *jia1_zhou1* (*California*) would be aligned with $\emptyset$.

If we repeat the generation and alignment process in §4.2 with the language L, then we would benefit from knowing the L word(s) associated with each frame component. Hence, we would be able to align *california* with *jia1_zhou1*.

Note that this approach is feasible only if

- G_L is also available
- The paraphrase in L, i.e., GEN(PARSE(S_L, P_L), G_L), is identical to S_L.

Key	$w_{Chinese}$	$w_{English}$
`weekday`	*ming2_tian1*	*tomorrow*
`:auxil`	*hui4*	*will*
`:verb`	*xia4_yu3*	*rain*

Table 4.1: Induced translation lexicon for JUPITER.

4.4 Translation Lexicon Induction

The $P_{L'}$`.translations` is induced based on the word alignments. Some translation lexicon entries inferred from Figure 4-2 are shown in Table 4.1.

4.4.1 Limitations

Note that the same L word might have multiple translations. For example, in a different sentence in the training set, the algorithm learns that an alternative translation of *hui4* under the `:auxil` key is *going to*. We keep a count of the number of occurrences of each translation variant, and pick the most frequently occurring one.

A statistical approach might be warranted to learn how the context disambiguates *will* and *going to*. In general, we would like to calculate $P(w_{English}|w_{L'}, key, context)$, where *context* may include, for instance, the other key-value pairs in the frame, or the name of the parent frame.

Chapter 5

Tree Transformation

We will illustrate the tree transformation steps with our running example, S_L = *Will it rain tomorrow*, with its parse tree T_L in Figure 3-2, and $S_{L'}$ = *ming2_tian1 hui4 xia4_yu3 ma5*. These steps depend on two pieces of information: the L-L' word alignment, and the structure of $S_{L'}$, which is gleaned from its generation process.

The output of the alignment process on S_L and $S_{L'}$ is shown in Table 5.1. In general, one could classify an alignment as one of five types, depending on whether there is zero, one or many words on either side of the alignment. In many languages, there is no consensus on what constitutes 'one' word. In our context, the number of words in a sentence may be defined as the number of lexicon look-ups (see §3.3.2) during its generation process in GENESIS. The five alignment types are shown in Table 5.2. Note that there is no $many_{L'}$ column, because, in practice, GENESIS uses features (see §3.3.1), rather than multiple look-ups, to generate semantically composed words. These words are then concatenated with underscores and generated at the same time.

For example, while *younger sister* is considered two words, its Chinese equivalent,

L word		L' word	Alignment type
will	$\longleftrightarrow^1$	*hui4*	one_L-$one_{L'}$
it	$\longleftrightarrow^2$	$\emptyset$	one_L-$\emptyset_{L'}$
rain	$\longleftrightarrow^3$	*xia4_yu3*	one_L-$one_{L'}$
tomorrow	$\longleftrightarrow^4$	*ming2_tian1*	one_L-$one_{L'}$
$\emptyset$	$\longleftrightarrow^5$	*ma5*	$\emptyset_L$-$one_{L'}$

Table 5.1: Word alignments between S_L and $S_{L'}$, and their types.

Type	$\emptyset_{L'}$	$one_{L'}$
$\emptyset_L$	n/a	insert (§5.4)
one_L	prune (§5.2)	replace (§5.1)
$many_L$	prune (§5.2)	choose (§5.2.1)

Table 5.2: Types of L-L' word alignments.

mei4_mei5, is considered one word because it is generated by one lexicon look-up on `sister` with the `$younger` flag.

We now describe how each type transforms the L parse tree.

5.1 Leaf Replacement

The most straightforward cases are one_L-$one_{L'}$ and one_L-$many_{L'}$ alignments, where we simply replace the leaf for the L word with its aligned L' word(s). This transformation handles alignments 1, 3 and 4 in Table 5.1. The resulting parse tree is shown in Figure 5-1.

5.2 Branch Pruning

For one_L-$\emptyset_{L'}$ and $many_L$-$\emptyset_{L'}$ alignments, we prune the branch(es) corresponding to the L word(s). For example, due to alignment 2 in Table 5.1, the branch for *it* in the tree in Figure 5-1 is pruned.

If the pruned branch contains a node that constructs a component in the semantic frame, that component would be lost. For example, the semantic frame produced by the proposed parse tree in Figure 5-1 would lack the `:topic` frame for the pronoun *it*, because the `subject` and `it` nodes are missing. The effects of branch pruning will be evident in our experiments on Chinese-to-English translation.

5.2.1 Choice for Pruning

Similarly, a pruning decision has to be made for a $many_L$-$one_{L'}$ alignment. Out of the many L branches, only one could remain and have its leaf replaced by the aligned L' word. The rest must be pruned.

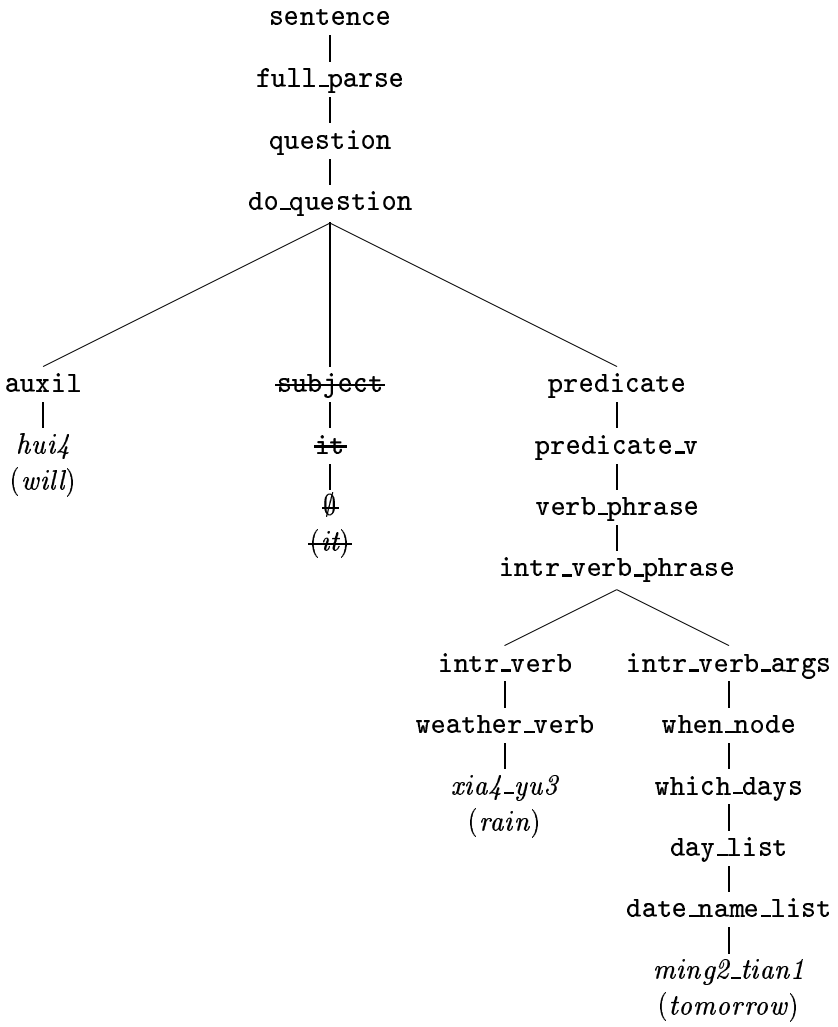


Figure 5-1: The proposed $T_{L'}$ after Leaf Replacement (§5.1). The leaves of T_L in Figure 3-2 were replaced with their aligned L' words. At Branch Pruning (§5.2), the `subject` branch, whose leaf is aligned with $\emptyset$, is pruned.

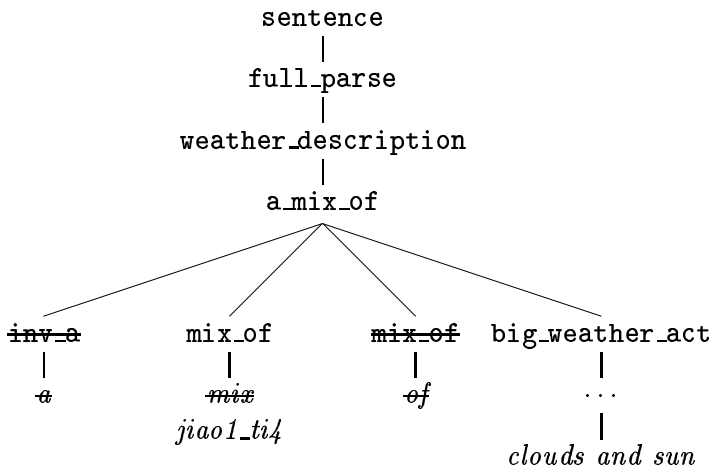


Figure 5-2: Due to a $many_L$-$one_{L'}$ alignment, *a mix of* $\longleftrightarrow$ *jiao1_ti4*, either the `inv_a` or the `mix_of` branch is to be pruned. A majority decision prunes the former.

Some of the L branches might have a non-null label[1] given by P.`actions` (see §3.2.3). This label indicates that the branch constructs a component in the frame, and is therefore presumably important to preserve. If there is only one such branch, we keep that branch and prune the rest.

There are two exception to this general principle. The first applies when a node with the instruction to create a `topic` has an ancestor with the same instruction. Unless it has priority[2] over the ancestor node, it is 'overridden' and its instruction ignored. Hence, it has no semantic importance.

The utterance *Any weather advisories in Boston* in Figure 5-3 illustrates this situation. The words *weather advisories* are aligned to a single L' word, *zai1_hai4_tian1_qi4_jing3_bao4*. Although `weather` has the instruction to create a `topic`, it is overridden by `an_advisory`, which has a higher priority for topics. The algorithm hence should not keep the `weather` branch simply because it has a non-null label from P.`actions`.

The second exception applies when a node is designated a `named_topic`, which means that it produces a `topic` frame with the key `:name` in it. The value of `:name` consists of the yield of the nodes, except those that are covered by another node with a non-null instruction.

For example, in Figure 5-3, the word *advisories* alone should be aligned to the `:name`

[1]Except the special `.invisible` label, which dictates that the node is irrelevant to the semantic interpretation.

[2]By virtue of the ordering in the `.actions` file.

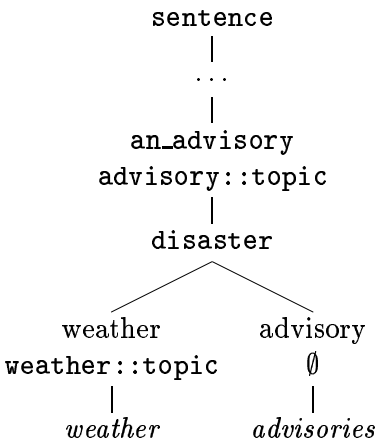


Figure 5-3: The `topic` instruction does not give the `weather` node any semantic importance, because it is overridden by `an_advisory`. The `advisory` branch should be kept because it contributes to the `:name` key in the `advisory` topic frame.

key in the `advisory` topic, because *weather* is covered by another topic-creating node.

The general principle does not suffice when there is none, or more than one such branch. In this case, we keep the branch whose label given by P_L.`rules` appears most often among the branches under consideration. The utterance *A mix of clouds and sun*, taken from the WEATHER RESPONSES domain, and its parse tree are shown in Figure 5-2. Its Chinese paraphrase is *yun2 he2 tai4_yang5 jiao1_ti4*. This pair of utterances has a $many_L$-$one_{L'}$ alignment, *a mix of* $\longleftrightarrow$ *jiao1_ti4*. Since the label `mix_of` appears twice, it is preferred over the label `inv_a`. We hence prune the latter, and keep the former and replace its leaf with *jiao1_ti4*.

5.2.2 Limitations

When there is a tie on the number of appearances of the labels, we take the first label. Although this heuristic generally works well for our test domains, we suspect it is a language-dependent issue. The proper solution to this problem, it seems, is to combine the nodes of the branches. As an example, for the alignment *younger sister* $\longleftrightarrow$ *mei4_mei5*, one might imagine putting the `relative_age` node under the `relationship` node in Figure 5-4, and indicating that it is to construct a `relative_age` predicate under the `relationship` topic frame. This would necessitate a new feature in P.`actions`.

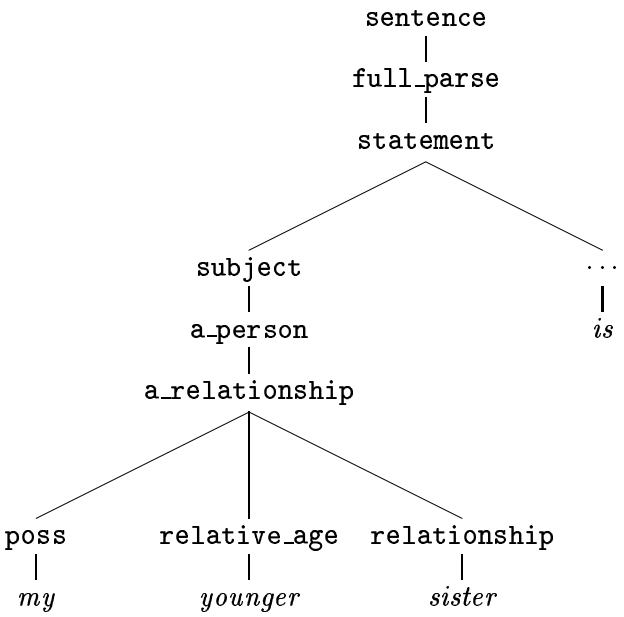


Figure 5-4: An example $many_L$-$one_{L'}$ alignment: both *younger* and *sister* map to one L' word, *mei4_mei5*.

5.3 Branch Movement

After replacing the L leaves with their L' translations, we re-order the branches to conform to the L' word order. The resulting parse tree is shown in Figure 5-5. This tree is by itself incorrect in at least two ways. It suggests that

- a `verb_phrase` node could be rewritten as a `when_node`, which contains no verb.
- a `do_question` node could be rewritten as two `predicate` nodes, with an `auxil` sandwiched between.

Pragmatically, this tree is also undesirable because it would place the `temporal` predicate directly under the `verify` frame, rather than under the `rain` predicate.

We will address these issues with the trace mechanism, described in Chapter 6. This naive branch movement is executed to facilitate the next transformation step, branch insertion.

5.4 Branch Insertion

When there is a $\emptyset_L$-$one_{L'}$ or $\emptyset_L$-$many_{L'}$ word alignment, a new branch has to be inserted into the tree. While the L' word order fixes the horizontal location of this new branch, it is

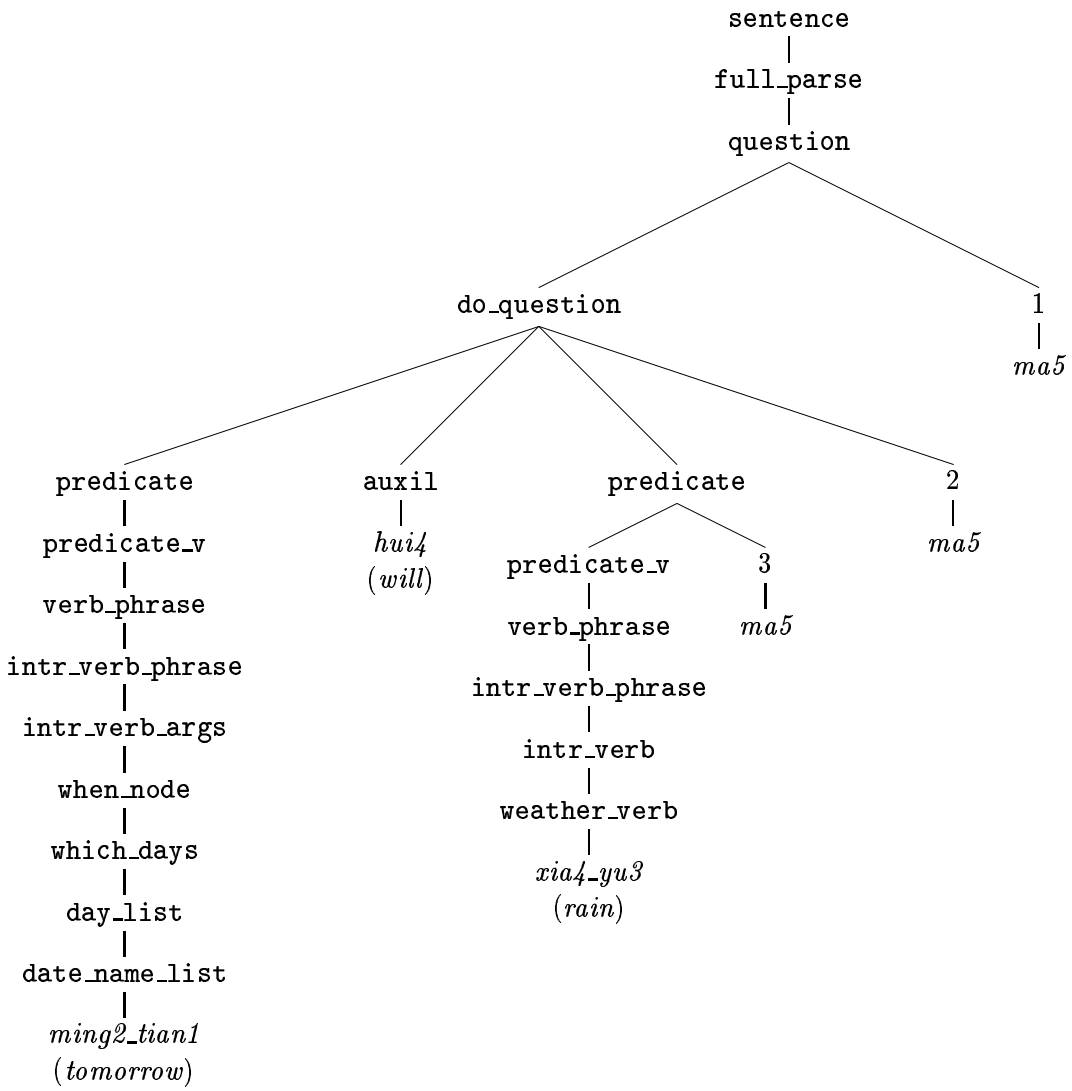


Figure 5-5: The proposed parse tree for $S_{L'}$, after Branch Movement (§5.3). The branch for *ming2_tian1* is moved to the front to conform to the L' word order. At Branch Insertion (§5.4), the branch for *ma5* may be inserted at three different locations. Location 1 is the most appropriate one.

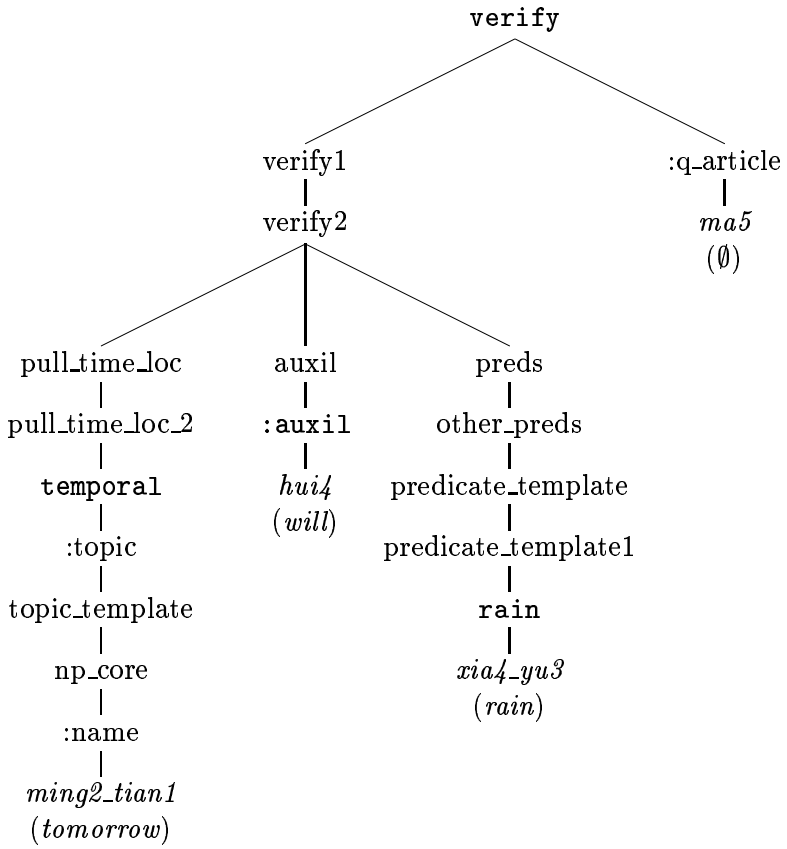


Figure 5-6: Generation tree for $S_{L'}$, yielding the bracketing (((*ming2_tian1*) (*hui4*) (*xia4_yu3*)) (*ma5*)).

unclear at which vertical location it should be merged with its neighbors.

In our running example, the alignment $\emptyset \longleftrightarrow ma5$ necessitates a branch insertion. As shown in Figure 5-5, there are three possible merge locations, all of which preserve the Chinese word order. We choose the most appropriate location on the basis of the structure of $S_{L'}$, which could be inferred during its generation process.

5.4.1 $S_{L'}$ bracketing

The generation rules in GENESIS (see Table 3.4) may be viewed as context-free rules. The non-terminals are either a frame name (e.g., `verify`), or a step name (e.g., `pull_time_loc`); the terminals are the surface strings produced by the lexicon. The generation process of $S_{L'}$ could thus be mapped to a 'generation tree', as shown in Figure 5-6. We interpret this tree as the structure of $S_{L'}$.

In general, we should insert a new branch into the parse tree in Figure 5-5 such that

the result would have the "same" structure as the generation tree. More concretely, the siblings of the inserted branch should have the same *yields* as the siblings of the branch in the generation tree.

In Figure 5-6, the sibling of the *ma5* branch is `verify1`, whose yield is *ming2_tian1 hui4 xia4_yu3*. If the *ma5* branch is inserted at location 1 in Figure 5-5, its sibling would be `do_question`, which has the same yield. However, if it were inserted at location 2, its sibling would be `predicate`, `auxil`, `predicate`, each with different yields. Location 3 would also result in different yields. The appropriate merge location, therefore, is 1.

A more intuitive way to describe the branch insertion algorithm is through bracketing. A parse tree may be converted into a bracketed sentence. For example, the trees in both Figure 5-6 and Figure 5-5 (if location 1 is taken) may be converted to (((*ming2_tian1*) (*hui4*) (*xia4_yu3*)) (*ma5*)).

Thus, the new branch should be inserted such that the resulting parse tree and the $S_{L'}$ generation tree would have the same bracketing. For example, if the bracketing from the generation tree were ((*ming2_tian1*) (*hui4*) (*xia4_yu3*) (*ma5*)), location 2 would have been the correct choice; if the bracketing were (((*ming2_tian1*) (*hui4*)) ((*xia4_yu3*) (*ma5*))), location 3 would have been the correct choice.

Chapter 6

Trace Simulation

After the Branch Insertion step (§5.4), the proposed parse tree $T_{L'}$ (see Figure 5-5) contains all the words in $S_{L'}$ in the correct order. However, the problems caused by the naive branch movement in §5.3 still remain. In this chapter, we address these problems using TINA's trace mechanism.

6.1 Trace Simulation

As discussed in §3.2.2, the trace mechanism in TINA is implemented as an implicit partnership among *generators*, whose constituents are allowed to move; *activators*, which license movement of the generated constituent; and *absorbers*, the designated destination of the generated constituent. This mechanism depends crucially on semantic constraints to restrict the kind of movements allowed. If these constraints are not carefully devised, the mechanism would lead to overgeneration.

The semantic constraints are specified in the form of semantic category assignments for the generators, activators and absorbers. These assignments are difficult to induce without substantial world knowledge. We therefore make the following simplifications, and leave it to the developer to implement the full trace scheme[1]:

- There is one generic *generator*, which has the general label `extraposed`. *The developer will need to classify this general label into appropriate semantic categories.*

[1]During evaluation, the grammars are automatically induced and are not further edited by humans. To accomodate the incomplete trace constraints of the induced grammars, a few adjustments were made to the parser. The details are described in §7.3.

- There is one generic *activator*, which has the general label `extrapose`. *Similarly, the developer will need to specify semantic restrictions for this general label.*

- In place of the special trace tag ⋆`trace`⋆, we use a keyword, ⋆`hyptrace`⋆, to denote a trace. It behaves like a normal word, and appears in the L' paraphrase when TINA and GENESIS are run in the grammar induction mode[2]. For example, S_L would be paraphrased as *ming2_tian1 hui4 xia4_yu3* ⋆`hyptrace`⋆ *ma5*. The ⋆`hyptrace`⋆ indicates that one of the preceding words has been extraposed. *This helps the developer locate places where traces are necessary.*

Since nodes are not assigned any semantic categories, there are no semantic constraints. In other words, a branch with an `extraposed` node would be moved to the closest branch to its right that has a ⋆`hyptrace`⋆ leaf.

6.2 Trace Detection Algorithm

6.2.1 Terminology

We start with a few definitions to facilitate our discussion:

- Suppose that S_L is of length N, with the words $w_1 w_2 w_3 \ldots w_N$; and that $S_{L'}$ is of length N', with the words $w'_1 w'_2 w'_3 \ldots w'_{N'}$.

- The *head node* of a set of nodes is the root node of the smallest subtree that contains all the nodes in the set. For example, in Figure 6-3, the head node of *hui4* and *xia4_yu3* is `do_question`; the head node of *xia4_yu3* and *ming2_tian1* is `intr_verb_phrase`.

- The *constituent node* of a word w_i is one of the following two nodes, whichever is the root of a smaller subtree. (1) the head node of the two leaf nodes corresponding to the words w_{i-1} and w_i; or (2) the head node of the two leaf nodes corresponding to w_i and w_{i+1}. For example, in Figure 6-3, the constituent node of *xia4_yu3* is `intr_verb_phrase`.

- The *constituent* of a word w_i is the yield of the constituent node of w_i. Let $\text{CONSTITUENT}_L(w_i)$ be the constituent of the word w_i in the L parse tree. For example, in Figure 6-3,

[2] See Appendix A.

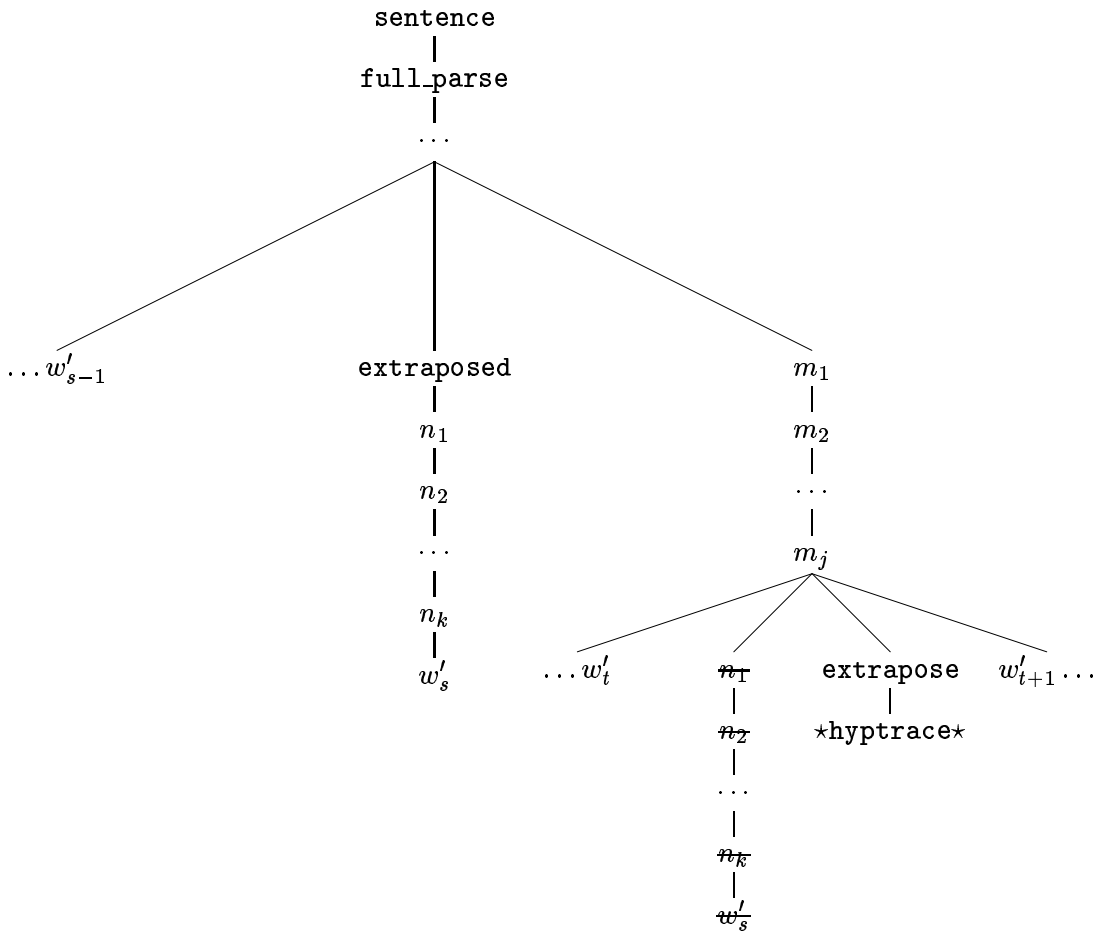


Figure 6-1: Schematic diagram of a typical trace movement. The word w'_s is extraposed from the position between w'_t and w'_{t+1}. Of the nodes taken from the branch of $\text{ALIGN}_L(w'_s)$ in the L parse tree, only $n_1 \cdots n_k$ should be moved forward to the extraposed branch. The nodes $m_1 \cdots m_j$ should not appear in that branch.

the constituent of *xia4_yu3* is *xia4_yu3 ming2_tian1*. Generally speaking, if $w_j \in \text{CONSTITUENT}_L(w_i)$, then, w_i and w_j are likely to appear within the same subframe in the semantic frame.

- Let $\text{ALIGN}_{L'}(w_i)$ be the $S_{L'}$ word to which w_i is aligned. If w_i is unaligned, then $\text{ALIGN}_{L'}(w_i)$ is $\emptyset$. Similarly, let $\text{ALIGN}_L(w'_i)$ be the S_L word to which w'_i is aligned.

6.2.2 Algorithm

Figure 6-1 shows a typical trace movement. The surface word w'_s (in our example, *ming2_tian1*) has been extraposed from its original position, between w'_t and w'_{t+1}. At parse time, we would like to place it back under an `extrapose` node between w'_t and w'_{t+1}.

```
[1]for i = N' to 1
    [2]if ALIGN_L(w'_i) ≠ ∅
        [3]for each w ∈ CONSTITUENT_L(ALIGN_L(w'_i))
                [4]if ALIGN_L'(w) ∉ CONSTITUENT_L'(w'_i)
                    Insert trace node (§6.2.3)
                    Edit extraposed branch (§6.2.4)
                end if
        end for
    end if
end for
```

Table 6.1: Trace detection algorithm

After the Branch Movement step (§5.3), all extraposed words are already in their surface position. What we need to do, then, is the following:

1. Identify t, i.e., the original location of the extraposed words, and insert ⋆`hyptrace`⋆.

2. Identify s, i.e., the words that have been extraposed, and insert an `extraposed` node in its branch. Furthermore, the Branch Movement step has copied the entire path of w'_s, taken from root to leaf in the L parse tree, to its new location. As illustrated in Figure 6-1, the nodes $m_1 \ldots m_j$ should not be in this new branch. Hence, we need to edit the extraposed branch.

One of the main objectives of the trace mechanism is to standardize the hierarchy of the subframes in the semantic frame. As discussed above, words in the same constituent tend to form components within the same subframe. Our premise is that if w_j is in the same constituent as w_i in the S_L parse tree, then $\text{ALIGN}_{L'}(w_j)$ and $\text{ALIGN}_{L'}(w_i)$ should also be in the same constituent in the $S_{L'}$ parse tree. If not, then one of them is considered to have been extraposed.

The Trace Detection algorithm is presented in pseudocode in Table 6.1. It traverses $S_{L'}$ from right to left (1). At every word w'_i, the algorithm computes its constituent in the L parse tree. If w'_i is not aligned to any L word, e.g., *ma5*, then the algorithm moves on to *xia4_yu3* (2).

The algorithm then examines every member of the constituent (3). In our example, this constituent consists of *rain* and *tomorrow*. The aligned L' word of each member is expected

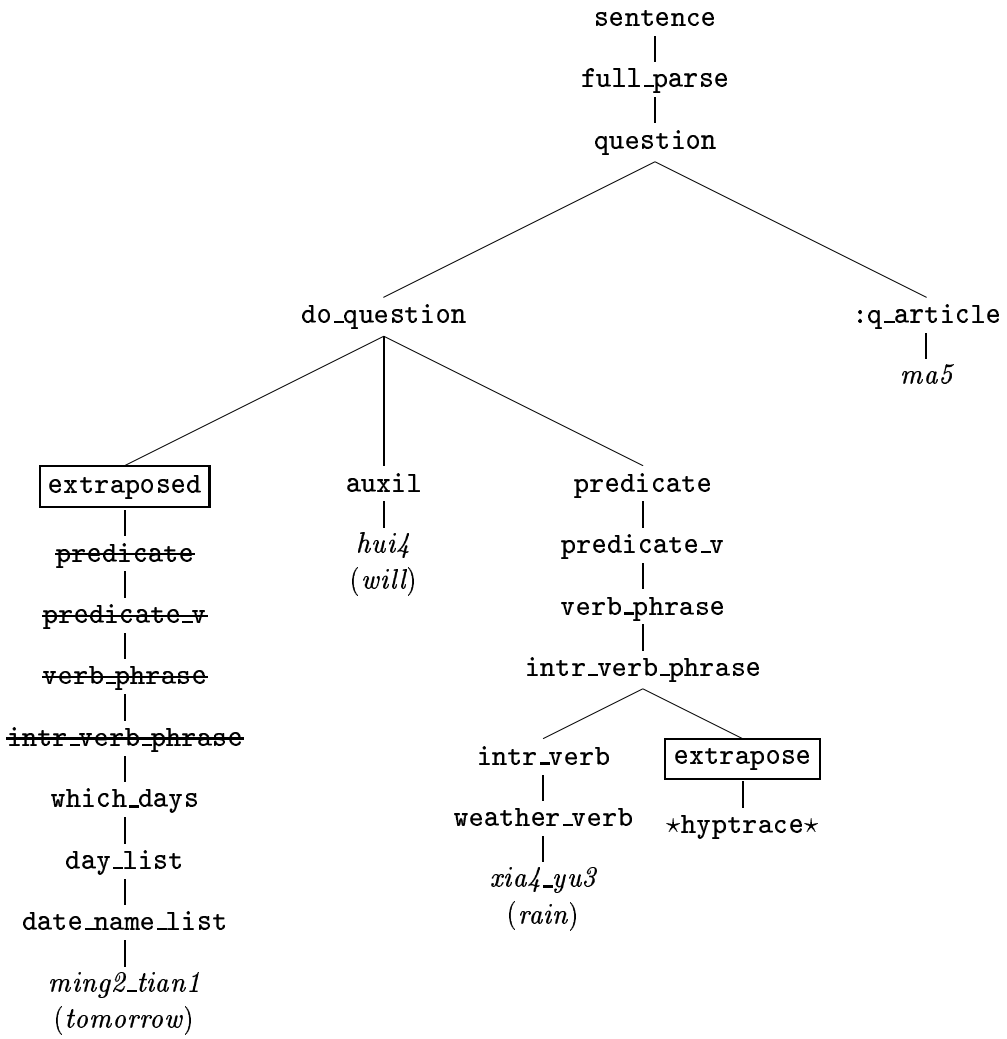


Figure 6-2: Proposed parse tree for $S_{L'}$, after applying the Trace Detection Algorithm (§6.2). At run time, the branch beneath `extraposed` would be cut off and grafted onto the `extrapose` node.

to be in the constituent of w'_i in the proposed L' parse tree (4). If not, it is considered to have been extraposed. In our example, the aligned L' word of *tomorrow*, *ming2_tian1*, is not in the same constituent as *xia4_yu3*. Hence, the operations in §6.2.3 and §6.2.4 are triggered.

6.2.3 Insert Trace

We insert the activator `extrapose` and the trace word ⋆`hyptrace`⋆ at the location from which the word w'_s has been extraposed.

We first find $\text{CONSTITUENT}_L(\text{ALIGN}_L(w'_s))$. In our example, the constituent of *tomorrow*,

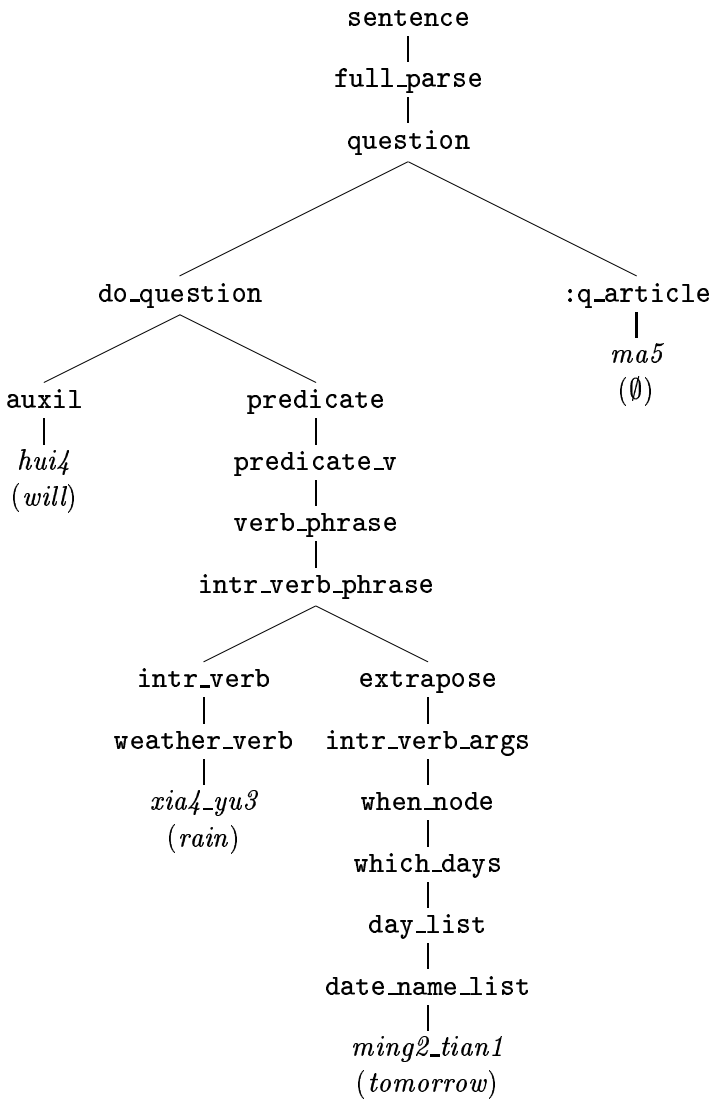


Figure 6-3: Proposed parse tree for $S_{L'}$, after its trace has been moved.

to which *ming2_tian1* is aligned, is `intr_verb_phrase`. Under this node, we insert an `extrapose` node, with a ⋆`hyptrace`⋆ as leaf, as in Figure 6-2. When the trace is moved during parsing, ⋆`hyptrace`⋆ absorbs the extraposed branch.

6.2.4 Edit Extraposed Branch

The algorithm first determines the head node of the nodes between w'_s and ⋆`hyptrace`⋆. In our example, it is the node `do_question`. The generator `extraposed` is inserted under this node. Further, we insert the nodes under $\text{CONSTITUENT}_L(\text{ALIGN}_L(w'_s))$, corresponding to the nodes $n_1, \ldots, n_k$ in Figure 6-1. The resulting tree is shown in Figure 6-2.

When the trace has been moved, the normalized parse tree should look like Figure 6-3.

6.3 Limitations

Without semantic categories, it is impossible to handle *nested* traces. Consider the example *(Which hospital)*$_1$ *was (Jane)*$_2$ *taken* (t_2) *to* (t_1), taken from [18]. There are two traces in this sentence: first, *which hospital* is generated, and is eventually absorbed by (t_1), which takes an object modified by the preposition *to*. Nested within this trace, *Jane* is generated, and is absorbed by (t_2), which absorbs the direct object of the verb *taken*. This nested trace would not have been feasible if the two traces were not distinguished by their different semantic categories.

Chapter 7

Evaluation

We evaluated our translingual induction approach outlined in previous chapters on two domains and two target languages. A Chinese grammar was induced from a hand-crafted English grammar in the JUPITER [26] domain; a French grammar was induced from a hand-crafted English grammar in the PHRASE BOOK [19] domain.

7.1 Evaluation Domains and Languages

JUPITER, a weather domain, is our group's most mature domain to date. Its English parsing grammar, P_e, and its Chinese generation grammar, G_c are well developed and tested. They are thus ideal as the prior information to our induction algorithm, and as the "gold standard" in the evaluation. Having been deployed on a public phone line for many years, the domain boasts a large data set available for training. Table 7.1 lists some example utterances in the domain.

i would like to know the weather in washington D C for the weekend
tell me about cities in california
thank you bye bye
what about dallas
what will the temperature be in los angeles california tomorrow october seventh
i am finished
nothing thanks
what is the wind speed in boston
is it going to be cloudy in new york tomorrow
is there any severe weather

Table 7.1: Example utterances in the JUPITER domain.

i'm british
i'm having trouble understanding you
how many days do you plan to stay
how much would it cost in american dollars
i want to exchange traveler's checks into local currency
where can i watch television
please help me
what bean curd dishes do you have
turn left at the next light
i have reserved a table for three people for six p m

Table 7.2: Example utterances in the PHRASE BOOK domain.

PHRASE BOOK is part of an on-going project aimed at providing translation assistance for travelers, and for students of a foreign language. Its nature requires that multilingual grammars be written rapidly. It is on domains like this that we hope to be able to apply our induction approach. Recently, a French generation module G_f was developed.

Compared to JUPITER, PHRASE BOOK covers a broader range of concepts, and has less data available for training. We felt that it would be a challenging domain and would provide complementary evaluation for our induction approach.

7.2 Evaluation Metric

Suppose we take a grammar for $L = e$(nglish) as prior information to induce a grammar for $L' = c$(hinese). Ideally, we should measure the quality of the induced P_c against a handcrafted P_c. Our group has developed MUXING [24], a Chinese version of JUPITER. However, the style of the semantic frames used in MUXING was found to be substantially different from JUPITER. Since the induced grammar aims to imitate JUPITER, MUXING would not be a good point of reference.

We decided to compare P_c against P_e in their ability to map two equivalent sentences, S_e and S_c, to the same semantic frame. This metric rewards the induction algorithm for producing semantic frames that are as close as possible to those in the source language. It thus has the long-term effect of encouraging uniformity across all languages in the domain.

We used P_e and the induced P_c, respectively, to produce two semantic frames, say F^e = PARSE(S_e, P_e) and F^c = PARSE(S_c, P_c). If P_c were perfect, then F^e and F^c should be identical.

A possible metric, then, would be to compare F^e and F^c via, for example, the accuracy

of the frame names, and the precision/recall of the key-value pairs. However, we felt that this metric would not be informative for those who are not familiar with our semantic frame. Moreover, for the travelers and students who use PHRASE BOOK, it is not the accuracy of semantic frame, but ultimately the quality of the translation, that matters most. Therefore, we decided to measure the accuracy of the English paraphrase of F^c.

We then obtained the English paraphrases, GEN(F^c, G_e) and GEN(F^e, G_e). The latter was assumed to be correct and served as the "gold standard". We calculated the word error rate of GEN(F^c, G_e) with respect to this gold standard. Note that, if PARSE failed to parse S_c, then GEN(F^c, G_e) is considered to be the empty string and hence given a 100% deletion error.

In the experiment with PHRASE BOOK the same metric was used, with $L' = f(\text{rench})$. Since the coverage of G_e for this domain was still limited, we evaluated F^f by comparing the French paraphrases GEN(F^f, G_f) and GEN(F^e, G_f). This evaluation provides additional insights into the translation challenges in a language other than English.

7.2.1 Limitations

For machine translation in broad domains, such as news articles, the current prevailing metric is BLEU [15]. BLEU requires multiple reference translations, and reports on a modified form of N-gram precision, with a penalty for sentence brevity.

We did not use BLEU as our metric for several reasons. First, we simply do not have access to multiple English paraphrases of F^e. Second, we would like to be conservative in our evaluation, given that the translation in PHRASE BOOK would be used as teaching material. Also, in restricted domains like JUPITER and PHRASE BOOK, a sentence does not exhibit as much nuance, nor as wide a range of possible translations as in broad domains for which BLEU is designed.

Nonetheless, the word error rate is a rather harsh measure for translation quality. For example, the following pair of GEN(F^c, G_e) and GEN(F^e, G_e) incurred a 43% error rate.

> What is *the* temperature in England *tomorrow*?
>
> What is temperature *tomorrow* in England?

In many cases different phrase orderings resulted in high error rates in paraphrases that were entirely acceptable. Hence, the error rate should be treated as an upper bound on the

actual performance.

7.3 Parser Preparation

As described in §6.1, the induced grammar gives only a subset of the necessary trace specifications. For example, it does not define any semantic constraints. Normally, a developer is expected to fill in the gaps. For evaluation purposes, however, we would like to avoid human intervention.

Hence, we enabled TINA to handle the induced form of traces by the following:

- The node `extrapose` is dynamically identified as an activator.
- The word ⋆`hyptrace`⋆ is dynamically identified as a trace node that serves the same function as ⋆`trace`⋆.

Roughly speaking, these changes emulate a normal trace mechanism with no semantic constraints.

The trace simulation (§6.3) does not currently handle multiple, serial traces in a sentence. Consider the sentence *a few showers early then changing to snow showers overnight*, an example taken from WEATHER RESPONSES. Its Chinese paraphrase is *zao3_xian1 (early) you3 (∅) yi1_xie1 (a few) zhen4_yu3 (showers), guo4_hou4 (then), ye4_jian1 (overnight) zhuan3_cheng2 (change) zhen4_xue3 (snow showers).* Under the trace mechanism, *zao3_xian1* is extraposed and should be moved after *zhen4_yu3*; similarly, *ye4_jian1* is extraposed, and should be moved after *zhen4_xue3.*

One possible solution is to break complex sentences like this into two, leaving only one trace per sentence. However, in practice, most sentences in the test domains contain no more than one trace.

7.4 Experiments on JUPITER

7.4.1 Data Preparation

We have a large corpus of English utterances of naive users asking about weather over the phone. Since it is critical for the induction algorithm to learn from valid English parse trees and Chinese paraphrases, we filtered this corpus in three stages.

First, we filtered out sentences to which our English grammar, P_e, could not give a complete parse. Next, we obtained Chinese paraphrases for the remaining sentences using our Chinese generation module, G_c. Since this generation module was not yet entirely mature, we further tested the quality of these paraphrases. We filtered out those paraphrases that could not be parsed by a previously authored Chinese grammar[1]. Finally, we manually filtered out those paraphrases that translated only a part of the original English sentence.

At the end of the process, we were left with a little over 7000 sentences. The average length of sentences in the test set is 6.0 words. We randomly set aside roughly 10% of these sentences for testing.

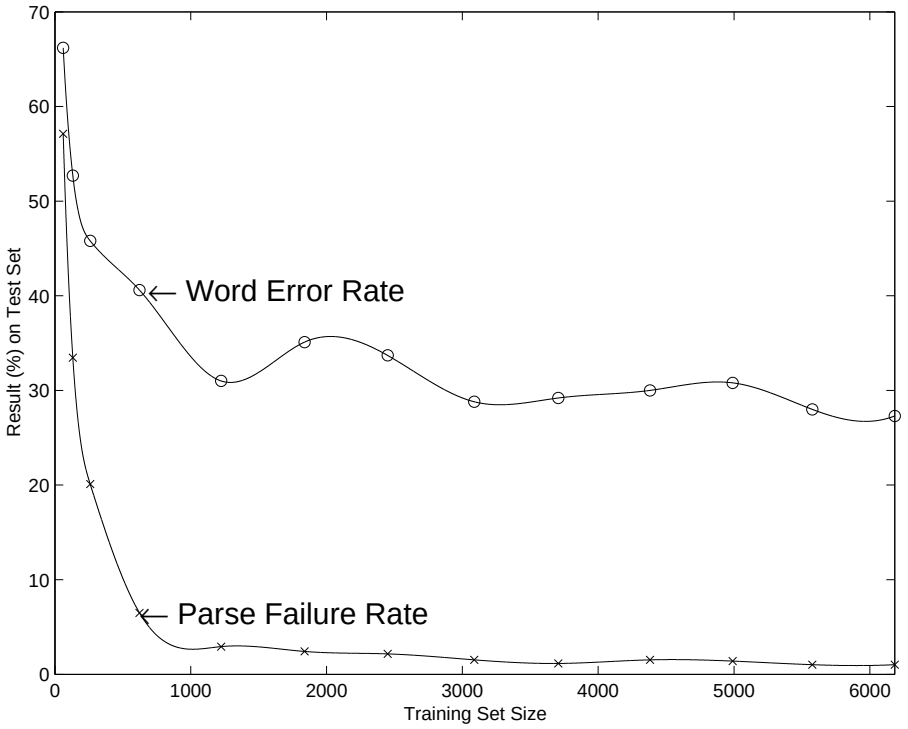


Figure 7-1: Performance of induced JUPITER grammar with respect to size of training data

7.4.2 Results

Figure 7-1 shows the learning curve of the induction algorithm. The best induced Chinese grammar performed at 27.3% word error rate (15.5% deletion, 7.7% substitution and 4.1% insertion rate) and 1.0% parse failure rate on the test set. Table 7.3 lists the 20 most

[1]In general, such a grammar of course would not pre-exist, and some other methods, such as manual assessment, would be required here.

Error	Word	%	Error	Word	%
del	*is*	9.6%	sub	*any* → *a*	1.7%
del	*the*	7.9%	sub	*is* → *does*	1.7%
sub	*for* → in	7.0%	del	*today*	1.6%
del	*be*	6.2%	sub	*will* → *is*	1.5%
del	*for*	4.8%	ins	*tomorrow*	1.5%
del	*it*	4.8%	del	*in*	1.4%
del	*like*	3.9%	sub	*how* → *what*	1.4%
ins	*about*	2.4%	del	*tomorrow*	1.3%
ins	*today*	2.0%	ins	*now*	1.3%
del	*will*	1.9%	sub	*are* → *is*	1.1%

Table 7.3: Twenty most frequent errors in the English paraphrases, GEN(F^c, G_e), as percentages of the total error. (del = deletion, sub = substitution, ins = insertion)

frequent errors, which collectively accounted for 65.0% of the total error.

7.4.3 Error Analysis

Almost all deletions of *tomorrow* or *today* in GEN(F^c, G_e) were coupled with insertions of the same words elsewhere in the paraphrase. This phenomenon illustrates the limitations of the word error rate as discussed in §7.2.1, and artificially increased the error rate.

Aside from *tomorrow* and *today*, none of the other words in Table 7.3 are content words that significantly alter the meaning of the paraphrase in the weather domain. The majority of the errors were deletions of words that were in fact absent from the Chinese paraphrases. The induction algorithm therefore pruned the branches of these words, leading the induced P_c to produce impoverished semantic frames. Such deletions would exist even for a grammar developed by an expert. It should be the responsibility of the pre-processor G_e.`pre` (see §3.3) to reinstate such missing features based on first principles and/or statistical methods.

Other errors were caused by translation variants of Chinese words in the domain (see §4.4.1). In the experiment, we simply selected the variant that was seen most often in the alignments. For instance, *zhi1_dao4* translates to the more frequently occurring *know about* rather than to *know*, accounting for most of the insertion errors for *about*.

Finally, mistakes in word alignment introduced some noise and redundancies to the grammar.

Error	Word	%
ins	*pas*	4.9%
sub	*voudrais* → *aime*	3.9%
del	*pas*	3.7%
del	*de*	3.0%
ins	*me*	2.0%
sub	*me* → <verb>	1.7%
ins	*un*	1.7%
sub	*combien* → *beaucoup*	1.5%
sub	*apporter* → *apporte*	1.5%
ins	*de*	1.5%

Table 7.4: Ten most frequent errors in the French paraphrases, GEN(F^f, G_f), in the training set. (See discussions in §7.5.3 for the errors involving <verb>.) The percentages are with respect to the total error. (del = deletion, sub = substitution, ins = insertion)

7.5 Experiments on PHRASE BOOK

7.5.1 Data Preparation

We have a corpus of just over 500 sentences in this domain, collected from students and research scientists in our research group. The average length of these sentences is 5.9 words. Using G_f, we obtained their French paraphrases.

7.5.2 Results

The induced French grammar performed at 14.0% (3.1% deletions, 7.7% substitution and 3.2% insertions) on the *training* set. Table 7.4 shows the ten most frequent errors.

7.5.3 Error Analysis

The insertions and deletions of *pas* were caused by mistakes in the word alignments. In French, negation is expressed by two words, *ne* and *pas*, which precedes and follows the verb. For example, the utterance *I do not eat meat* is translated as *Je(I) ne mange(eat) pas la viande(meat).* The induction algorithm erroneously aligned *ne* to *not*, and *pas* to $\emptyset$. During generation, *ne* and *pas* were both placed in front of the verb, resulting in *Je ne pas mange la viande*, with one deletion and one insertion of *pas.*

Incorrect word alignments were also responsible for the confusion between *beaucoup* and *combien.* The former was properly aligned to *much.* However, the latter, which means *how*

```
:pred {p bring
        :topic {q knife }
        :pred {p indir
                :topic {q pronoun
                          :name "me" } } }
```

Figure 7-2: The `bring` predicate in the semantic frame for *bring me the knife.*

much, was aligned only to *much*, resulting in ambiguity between the two words.

The words *vouloir* and *aimer* are French translation variants of *like*. The generation module generally maps *like* to *aimer*, except when it is preceded by *would*, in which case it maps *would like* to *vouloir*. Our induced grammar failed to learn this rule, and always translated *like* to *vouloir*, which was seen more frequently in the training data than *aimer*.

The French word *de* is a function word that could roughly be translated as *of*, as in *quels types de plats* (*what kinds of dishes*). In some expressions, such as *combien de jours* (*how many days*), it is not aligned to any English word, and is generated according to propagated features and frame contexts. For some sentences the induced grammar created frames that did not generate *de*, or inserted it at inappropriate positions.

A failure to detect constituent movement produced the errors involving <`verb`>, *me* and *apporter*. In French, an indirect object of the verb may be moved to the front of the verb. For example, the translation of *can you bring me a knife* is *pouvez (can) - vous(you) m' (me) apporter (bring) un couteau (knife).*

The semantic frame of the phrase *bring me a knife* is shown in Figure 7-2. The induced French grammar, not recognizing the movement, put the `indir` predicate on the same level as the `bring` predicate, resulting in an extra, ill-formed verb frame. This frame generated the <`verb`>, and also misplaced the *me* and conjugated *apporter* incorrectly.

Chapter 8

Future Plans

We plan to further our research in many directions, including:

8.1 Growing a Grammar

We anticipate that a developer will need to make adjustments to an induced grammar. After the developer has improved the grammar, s/he may later want to extend its coverage to more sentences in the domain. We would like to enable the induction algorithm to carefully add new induced rules to the modified grammar, while respecting the changes made by the developer.

8.2 Other Languages

We are presently developing generation modules for Mandarin, French, Japanese, Spanish and Korean in the PHRASEBOOK domain, intended for tourists who do not speak the language in their destination countries. In the future we plan to expand to Arabic and Urdu. This domain will be incorporated into our language learning system. All three languages used in our experiment, English, French and Chinese, are subject-verb-object languages. We would like to see the performance of this induction approach when L and L' have very different word ordering, such as English and Japanese.

8.3 Improvement on Word Alignment

The current algorithm is very sensitive to correct L-L' word alignment. If $G_{L'}$ is not yet working well, the quality of the induced grammar degrades significantly. A substantial amount of errors in our experiments directly or indirectly resulted from misalignments. A statistical treatment on word alignment may be warranted. A side benefit is that the induction algorithm may be useful for identifying generation errors.

8.4 Other Evaluation Metrics

While we have thus far only evaluated the induced grammar on paraphrases into *natural* languages, we are also interested in applying the grammar in spoken dialogue applications, where the system must understand the query and respond appropriately. We generally use a 'paraphrase' into a flattened (key: value) representation to transform the semantic frame into a format that is more transparent to the dialogue manager. Formal evaluation of the differences in this (key: value) representation could help us judge the effectiveness of our generated grammars for dialogue interaction.

Appendix A

Implementation Details

A.1 Induction Command

To induce a grammar, issue the command:

> `tina` $< D >$ `-language` L' `-frame -induce` L `-induceFile` $< path >$ `-sentences` $< trainfile >$

where:

- $< L' >$ is the target language
- $< L >$ is the source language
- $< D >$ is the domain
- $< trainfile >$ is the file containing training sentences in L
- $< path >$ is the directory in which the induced grammar files will be placed.

The induced grammar files include the following:

induced.trace.rules The rewrite rules in $P_{L'}$`.rules` (see §3.2.1).

induced.translations The key value translations in $P_{L'}$`.translations` (see §3.2.4).

induced.trace.sents The L' paraphrases of the sentences in $< trainfile >$, with hypothesized traces indicated by ⋆`hyptrace`⋆.

induced.constraints See description below.

induced.log This file shows the word alignment and rules induced from each training sentence.

A.2 Constraints File

The .constraints file is the same for all induced grammars:

```
.*list*
*generators*

.*generators*
extraposed
```

The purpose of this file is to declare `extraposed` as a generator.

A.3 Trace Simulation Command

To run TINA with the trace simulation (see Chapter 6), use the -sim_trace option. For example, to run the induced grammar, issue the command:

`tina` $< D' >$ `-language` L'' `-frame -sim_trace -sentences` $< path >$/induced.trace.sents

where:

- $< D' >$ is an appropriately defined domain for the induced grammar
- $< L'' >$ is the language in which the induced grammar is to be evaluated

In the -sim_trace mode, `extrapose` is declared an activator, and ⋆`hyptrace`⋆ declared a trace node, just before TINA starts parsing.

Bibliography

[1] L. Baptist and S. Seneff. GENESIS-II: A versatile system for language generation in conversational system applications. In Proc. Intl. Conf. on Spoken Language Processing, pages 271–274, Beijing, China, October 2000.

[2] E. Charniak. Tree-bank grammars. Technical Report CS-96-02, Brown University, Providence, RI, 1996.

[3] E. Charniak. A maximum-entropy-inspired parser. In Proc. of the First Language Technology Joint Conference of Applied Natural Language Processing and the North American Chapter of the Association for Computational Linguistics, Seattle, WA, April 2000.

[4] S. Chen. Bayesian grammar induction for language modeling. In Proc. Mtg. of the Assoc. for Computational Linguistics, pages 228–235, Cambridge, MA, 1995.

[5] N. Chomsky. Lectures on Government and Binding. Foris, Dordrecht, Germany, 1981.

[6] M. Collins. Three generative, lexicalized models for statistical parsing. In Proc. Mtg. of the Assoc. for Computational Linguistics, Madrid, Spain, 1997.

[7] B. Cowan. Tailoring meaning representations for machine translation using PLUTO. In Proc. MIT Student Oxygen Workshop, (http://sow.lcs.mit.edu/2003/proceedings.html), Gloucester, MA, September 2003.

[8] M. Gavaldà and A. Waibel. Growing semantic grammars. In Proc. Mtg. of the Assoc. for Computational Linguistics, Montréal, Canada, August 1998.

[9] R. Hwa. Supervised grammar induction using training data with limited constituent information. In Proc. Mtg. of the Assoc. for Computational Linguistics, College Park, MD, June 1999.

[10] H. Lasnik and M. Saito. Move Alpha: Conditions on its Application and Output. MIT Press, Cambridge, MA, 1994.

[11] A. Lavie, L. Levin, T. Schultz, C. Langley, B. Han, A. Tribble, D. Gates, D. Wallace, and K. Peterson. Domain portability in speech-to-speech translation. In Proc. Human Language Technology Conf., San Diego, CA, March 2001.

[12] C.N. Li and S.A. Thompson. Chinese: A Functional Reference Grammar. U. California Press, Berkeley, CA, 1989.

[13] M. Marcus, B. Santorini, and M. Marcinkiewicz. Building a lartge annotated corpus of english: the penn treebank. Computational Linguistics, 19(2):313–330, 1993.

[14] M. McCandless and J. Glass. Empirical acquisition of word and phrase classes in the ATIS domain. In Proc. Intl. Conf. on Spoken Language Processing, Yokohama, Japan, 1994.

[15] K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a method for automatic evaluation of machine translation. In Proc. Mtg. of the Assoc. for Computational Linguistics, pages 311–318, Philadelphia, PA, July 2002.

[16] F. Pereira and Y. Schabes. Inside-outside reestimation from partially bracketed corpora. In Proc. Mtg. of the Assoc. for Computational Linguistics, pages 128–135, Newark, DE, 1992.

[17] P. Price. Evaluation of spoken language systems: The ATIS domain. In Proc. DARPA Speech and Natural Language Workshop, pages 91–95, 1990.

[18] S. Seneff. TINA: A natural language system for spoken language applications. Computational Linguistics, 18(1):61–86, 1992.

[19] S. Seneff. Spoken conversational interaction for language learning. In Proc. InSTIL Symposium, Venice, Italy, June 2004.

[20] S. Seneff, C. Chuu, and D.S. Cyphers. ORION: From on-line interaction to off-line delegation. In Proc. Intl. Conf. on Spoken Language Processing, pages 142–145, Beijing, China, October 2000.

[21] S. Seneff, J. Glass, T.J. Hazen, Y. Minami, J. Polifroni, and V. Zue. MOKUSEI: A japanese spoken dialogue system in the weather domain. NTT R&D, 49(7), 2000.

[22] K.C. Siu and H.M. Meng. Semi-automatic grammar induction for bi-directional english-chinese machine translation. In Proc. European Conf. on Speech Communication and Technology, Aalborg, Denmark, 2001.

[23] A. Tribble, A. Lavie, and L. Levin. Rapid adaptive development of semantic analysis grammars. In Proc. Intl. Conf. on Theoretical and Methodological Issues in Machine Translation, Keihanna, Japan, March 2002.

[24] C. Wang, S. Cyphers, X. Mou, J. Polifroni, S. Seneff, J. Yi, and V. Zue. MUXING: A telephone-access mandarin conversational system. In Proc. Intl. Conf. on Spoken Language Processing, Beijing, China, 2000.

[25] A. Acero Y. Wang. Combination of cfg and n-gram modeling in semantic grammar learning. In Proc. European Conf. on Speech Communication and Technology, Geneva, Switzerland, September 2003.

[26] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington. JUPITER: A telephone-based conversational interface for weather information. IEEE Transactions on Speech and Audio Processing, 8(1):85–96, January 2000.

[27] V. Zue, S. Seneff, J. Polifroni, M. Phillips, C. Pao, D. Goodine, D. Goddeau, and J. Glass. PEGASUS: A spoken dialogue interface for on-line air travel planning. Speech Communication, 15(3-4):331–340, 1994.