A Hybrid Tree Framework for Semantic Parsing and Natural Language Generation

by
Lu Wei

M.Sc. in Computer Science
Singapore-MIT Alliance, National University of Singapore, 2006

SUBMITTED TO THE SMA OFFICE IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE (CS) AT THE
SINGAPORE-MIT ALLIANCE
2009

Signature of author: _____

CS Programme
June 30, 2009

Certified & Accepted by: _____

Assoc. Prof. Ng Hwee Tou
Thesis Advisor, SMA Fellow, NUS
Programme Co-Chair, Computer Science Programme

Certified by: _____

Assoc. Prof. Lee Wee Sun
Thesis Advisor, SMA Fellow, NUS

Certified by: _____

Prof. Leslie Pack Kaelbling
Thesis Advisor, SMA Fellow, MIT

Accepted by: _____

Prof. Tomas Lozano-Perez
Programme Co-Chair, Computer Science Programme

A Hybrid Tree Framework for Semantic Parsing and Natural Language Generation

by
Lu Wei

Submitted to the SMA office
In partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy in Computer Science (CS)

ABSTRACT

In this thesis, we present a novel framework based on *hybrid trees* that aims to bridge natural language sentences and their underlying meaning representations. The framework is guided by theoretical principles related to language and semantics. The purpose of the framework is to facilitate the development of systems that transform natural language sentences into their underlying meaning representations (called *semantic parsing*), as well as systems that transform meaning representations into their corresponding natural language sentences (called *natural language generation*).

Within the framework, we build a novel generative model that jointly generates both a natural language sentence and its meaning representation. We also develop efficient training and decoding algorithms for the proposed generative model. The generative model has a natural symmetry, allowing for easy transformation from natural language to meaning representation, and vice versa.

In practice, though the generative model gives reasonable performance, it still exhibits some limitations. To address these limitations, additional discriminative techniques are added to the generative model when performing both semantic parsing and natural language generation. We demonstrate through experiments that such a pipelined approach gives significant improvements in performance. In particular, the generative model, when augmented with discriminative techniques, outperforms previous state-of-the-art systems when evaluated on standard benchmark corpora.

Keywords: Hybrid Tree, Semantic Parsing, Natural Language Understanding, Natural Language Generation.

Thesis Advisors:

1. Assoc. Prof. Ng Hwee Tou, SMA Fellow, NUS
2. Assoc. Prof. Lee Wee Sun, SMA Fellow, NUS
3. Prof. Leslie Pack Kaelbling, SMA Fellow, MIT

To my parents

Acknowledgements

I am fortunate to have the privilege to work closely and have interaction with 3 excellent professors in the special and unique SMA programme: Prof. Ng Hwee Tou, Prof. Lee Wee Sun, and Prof. Leslie Pack Kaelbling.

Prof. Ng is an expert in the field of natural language processing. He helped me a lot on finding research topics and directions, contacting various other researchers for enquiries, pointing to relevant papers and researchers in the community. Prof. Ng is very good at finding good research directions. He has a very good view about the big picture in the field, which is very important. He is also extremely careful in academic writings and presentations. From him I have learned a lot.

Prof. Lee is very good at machine learning and statistical methods. We had lots of very useful discussions on how to build enhance statistical models effectively, how to improve algorithms and so on. He is very approachable and I can always hear many good technical suggestions from him during discussions and emails.

Although Prof. Kaelbling's main research focus was on machine learning and robotics, she is still a very responsible supervisor from whom I have learned a lot. She is very good at posing high level questions for my research. These questions can be very important but are often overlooked by me.

I would like to than Luke Zettlemoyer from MIT. During my visit to MIT, I had lots of useful discussions with him. He is very brilliant and shared with me lots of insightful views and provided me lots of suggestions on my research.

I would also like to thank Prof. Michael Collins from MIT. I took his course when I visited MIT. Through the course I have learned a lot. I also had a chance to talk to him about my research, and had received a strong encouragement from him.

My thanks also go to the following persons for their help throughout my studies: Prof. Kan Min-Yen, Prof. Tan Chew Lim, Sun Hongyang, Wu Dan, Dr. Na Seung-Hoon, Dr. Preslav Nakov, Daniel Dahlmeier, Tan Yee Fan, and many other friends and lab-mates from NUS.

Contents

List of Symbols and Abbreviations

Abbreviation	Description	Definition
MR	meaning representation	page 2
N-M pair	natural language - meaning representation pair	page 26
NL	natural language	page 2
NLG	natural language generation	page 3
NLP	natural language processing	page 1
SP	semantic parsing	page 2

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

One of the ultimate goals in the field of natural language processing (NLP), as an important subfield of artificial intelligence, is to enable computers to converse with humans through the media of human natural language. To achieve this goal, two important issues need to be studied. First, it is important for computers to accurately capture the complete underlying semantics of natural language sentences. Second, the computers should be able to accurately produce meaningful human-understandable natural language sentences from certain given semantics. These two tasks are usually referred to as *semantic parsing* and *natural language generation* respectively in the NLP field.

Traditional approaches to the above two problems are mainly manually constructed knowledge-based systems. An earlier example of such systems include the expert systems [HRWL83] that attempt to provide human-computer interactions. However, such systems require substantial human efforts to build and many of them are rule-based and are therefore not robust enough. Also, they are usually hard to port to new domains. Recently, statistical approaches became dominant in NLP and have been successfully applied to various shallow semantic analysis tasks, such as semantic role labeling [GJ02] and word sense disambiguation [NL96]. The task of building automated semantic parsing sys-

tems at a deeper level that are able to understand and generate human languages are still not widely studied [Moo04].

Wong [Won07] recently described in his thesis an unified framework for performing semantic paring and language generation. The framework was heavily motivated by the recent development of statistical machine translation systems. In such an framework, the semantic parsing and language generation tasks are both viewed merely as a parsing-based string-to-string translation task. However, the meaning representation usually comes with structures such as a tree structure. It is not clear if a string-to-string based translation approach can well capture the structural information of the meaning representation. In this thesis, we introduce a novel unified framework called a *hybrid tree* that bridges natural language and meaning representations. The framework is guided by some theoretical principles related to language and semantics, such that models for semantic parsing and language generation can be built on top of such a framework. We also introduce generative models based on the framework, and develop algorithms that can efficiently perform model parameter estimation as well as efficient decoding.

1.1 Semantic Parsing

Semantic parsing (SP) (or sometimes referred to as *natural language understanding* (NLU) in some literature [PRW97]) is the task of mapping natural language (NL) sentences into their corresponding formal meaning representations (MR). Such formal meaning representations can usually be used to perform further tasks by machines such as reasoning and inference.

For example, in one of the early work in the this field, Papineni et.al [PRW97] developed a system for semantic parsing in the Air Travel Information System (ATIS) [Hir92] domain. The example NL sentence

```
show me all the nonstop flights from city-1 to city-2
                leaving city-1 after time-1 on date-1
```

corresponds to a formal meaning representation of the following form:

```
LIST FLIGHTS NONSTOP DEPARTING AFTER TIME-1 FLYING-ON DATE-1
                FROM:CITY CITY-1 TO:CITY CITY-2
```

Such a meaning representation can be processed by the computer for various tasks. For example, retrieving information from an underlying database and returning desired answers to the users.

Semantic parsing has many potential applications in various NLP tasks. Examples include knowledge-based machine translation [NM92], document summarization [Man01], and question answering [NH04].

1.2 Natural Language Generation

Natural language generation (NLG) is the task of translating formal meaning representations to natural language sentences. Tasks that are studied in traditional NLG field are usually divided into two categories, namely strategic generation and tactical generation [Tho77]. Strategic generation refers to the task of making decisions on what meanings to express. Tactical generation refers to the task of effectively generating NL sentences given certain particular meaning representations, which is the inverse task of semantic parsing.

In this thesis, we focus on the tactical generation task. Natural language sentences are generated from their underlying meaning representations. For illustration purpose, we take an example from the GEOQUERY corpus [ZM96] that consists of queries regarding U.S. geographical information. The below meaning representation

$$answer(count(exclude(state(\text{all}), loc\_1(river(\text{all})))))$$

generates the following NL sentence:

```
How many states do not have rivers ?
```

Such a transformation process enables the computer to interact with human in a human natural language, and therefore is crucial in human-computer interaction applications [ABD+01]. Some other possible applications of NLG systems include indicative summarization [KMK01] and spoken dialogue systems [OR00].

1.3 Thesis Contributions

We list the contributions of this thesis, as follows:

1. This thesis introduces a novel framework called "*hybrid tree*" that bridges natural language and meaning representations. The framework, guided by certain theoretical principles relating language and semantics, is highly generic and in principle applies to all kinds of human natural language texts and all meaning representations that can be transformed into a binary tree structure.

2. This thesis also introduces a new recursive generative model built on top of the proposed hybrid tree framework. The generative architecture models a simultaneous generation process for both natural language and meaning representations. The generative process creates nodes for the hybrid tree at each level recursively. Such an elegant property provides the convenience of performing transformation task in both directions (*i.e.*, from natural language to meaning representation, and from meaning representation to natural language).

3. We also develop a novel dynamic programming algorithm for efficient training and decoding. The algorithm successfully reduced the inference time complexity from $O(n^6m)$ to $O(n^3m)$, where n refers to the length of the natural language sentence, and m is the number of tokens in the meaning representation. Though the algorithm is specifically designed for the proposed generative models, we believe the idea conveyed by such an algorithm can be helpful in other parsing tasks with similar scenario.

4. We also present methods that can be added to the existing trained generative model to achieve the tasks of semantic parsing and language generation. Experimental results show that our methods achieve comparable or even better performance compared to recent models designed for the same task, and outperform other state-of-the-art systems for both tasks. Evaluations were performed on standard publicly available corpora.

The works of semantic parsing and natural language generation with hybrid tree presented in this thesis are separately published in [LNLZ08] and [LNL09].

1.4 Thesis Overview

The rest of the thesis is organized as follows:

- Chapter 2 introduces the MR formalism and gives a comprehensive survey on previous research work in the field of SP and NLG;

- Chapter 3 formally introduces the hybrid tree framework. We introduce various concepts and notation used throughout this thesis;

- Chapter 4 introduces three generative models built on top of the hybrid tree framework. We also present the parameter estimation methods for efficiently training the model;

- Chapter 5 presents a proposed novel dynamic programming algorithm for efficient parameter estimation and decoding;
- Chapter 6 describes methods for semantic parsing and language generation under the hybrid tree framework, which are used together with the proposed generative models;
- Chapter 7 presents experimental results for the proposed model when applied to both SP and NLG tasks, together with discussions for comparisons with previous models;
- Chapter 8 gives some discussion about potential further research work;
- and Chapter 9 gives concluding remarks.

Chapter 2

Background and Related Work

This chapter gives some background knowledge that is required in order to read this thesis. We discuss various meaning representation (MR) formalisms in Section 2.1, with a specific focus on the MR formalism that is used throughout this thesis. We present surveys on previous approaches for the semantic parsing task and language generation task in Section 2.2 and Section 2.3 respectively.

2.1 Meaning Representation

Natural language (NL) sentences can be represented in a symbolic structural form, which is usually referred to as the meaning representation (MR). Such MRs are usually formally defined with canonical forms, and can help perform specific natural language processing tasks, such as NLU and NLG. There exist various proposals for MR formalisms. Examples include first-order logic (FOL), proposition logic, semantic networks etc. [JM08].

In particular, throughout this thesis, we restrict our meaning representation (MR) formalism to a variable free version as presented in [KWM05, WM06]. Specifically, the formalism defines MRs with tree structures. An example MR

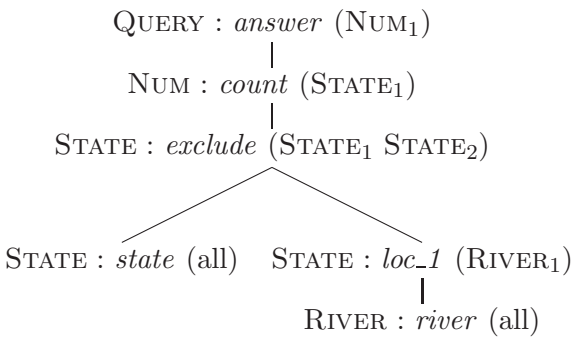


Figure 2.1: An example MR tree

tree is shown in Figure 2.1. This MR tree represents the following NL sentence:

```
How many states do not have rivers ?
```

which consists of 8 words, including the punctuation symbol "?".

The MR tree was originally converted from a Prolog query [KWM05]. Following a preorder traversal of this MR tree, we can equivalently represent it with the following list of *meaning representation productions* (MR productions):

(0) QUERY : *answer* (NUM$_1$)
(1) NUM : *count* (STATE$_1$)
(2) STATE : *exclude* (STATE$_1$ STATE$_2$)
(3) STATE : *state* (all)
(4) STATE : *loc_1* (RIVER$_1$)
(5) RIVER : *river* (all)

Each such MR production can be written in the following form:

$$\mathcal{M}_a : p_\alpha(\pi)$$

An MR production consists of three components: a *semantic category* $\mathcal{M}_a$, a *function symbol* p_α, as well as an *argument list* π.

The semantic category defines the meaning category that this MR production represents. The function symbol can be omitted (*i.e.*, empty). The argument list contains arguments, where an argument can be either a child semantic category or a constant.

Take production (1) for example: it has a semantic category "NUM", a function symbol "*count*", and a child semantic category "STATE_1" as its only argument. Production (5) has "RIVER" as its semantic category, "*river*" as the function symbol, and "all" is a constant in the argument list.

A string representation of an MR can be obtained by repeatedly substituting all MR productions' child semantic categories with their child productions. The above MR tree is equivalent to:

$$answer(count(exclude(state(\text{all})\ loc\_1(river(\text{all})))))$$

Another example MR is given in Figure 2.2, which is the MR of the following NL sentence: `If the ball is in our half then player 4 should be positioned at 47 meters from our goal with its ball attraction as -LRB- 0.9, 0.4 -RRB-.`

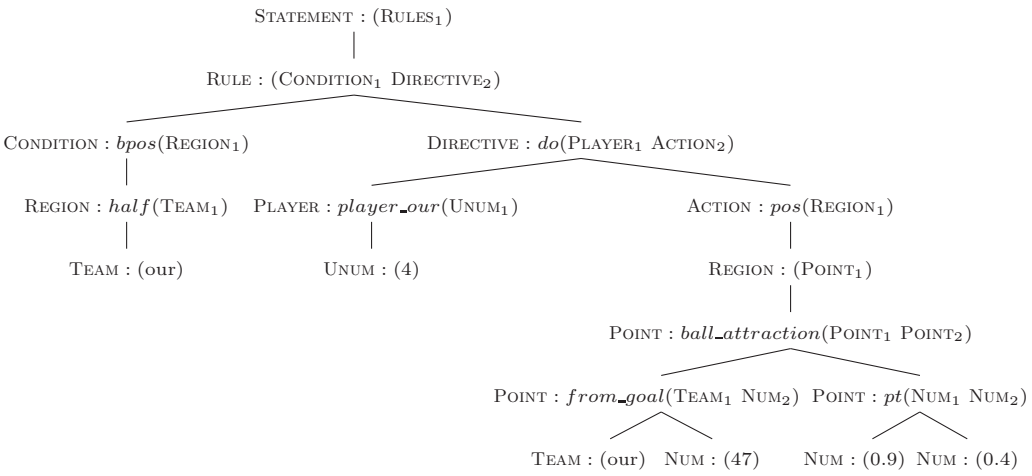


Figure 2.2: Another example MR tree

2.2 Semantic Parsing

There are many previous approaches for this task. Some earlier works in the field include several models built for the Air Travel Information System (ATIS) [Hir92] domain, where the meanings are represented with a frame-based structure that consists of attribute-value pairs: a model using hidden clumpings by Epstein el al. [EPR+96], the fertility models by Pietra et al. [PERW97], a maximum-entropy model by Papineni el al. [PRW97], as well as an alignment template model by Macherey et al. [MON01].

Recent systems that make use of the variable-free version of tree-structured functional meaning representation include the SILT [KWM05] system by Kate et al., the WASP [WM06] system by Wong and Mooney, the KRISP system and its extensions [KM06, KM07b, KM07a] by Kate and Mooney, as well as the SCISSOR system and its extensions [GM05, GM06] by Ge and Mooney. There are also some works that make use of lambda calculus as the meaning representation, which include the λ-WASP system [WM07b] by Wong and Mooney, as well as the works by Zettlemoyer and Collins [ZC05, ZC07].

2.2.1 The Silt System

The SILT (Semantic Interpretation by Learning Transformations) system [KWM05] is a transformation-based system which assumes that there exists a set of rules that transforms natural language to the corresponding MRs. The transformation rules are defined over a set of possible patterns extracted from the natural language sentences. The system presents two algorithms for performing such a semantic parsing task. The first version extracts plain strings as patterns from natural language sentences, while the second extracts trees (words or syntactic labels) as patterns from syntactic parse trees of the NL sentences.

We illustrate the algorithm with an example taken from their paper [KWM05], which makes use of the string pattern. Consider the following NL sentence:

`If our player 4 has the ball , our player 4 should shoot .`

To illustrate, assume there exist the following string-based rules that can be applied:

`our`	$\Rightarrow$	TEAM : (our)
`player 4`	$\Rightarrow$	UNUM : (4)
`shoot`	$\Rightarrow$	ACTION : (shoot)

These rules mean that a transformation can be made if there is a matching pattern that appears to the left of $\Rightarrow$ sign. The matched pattern shall be replaced with the MR production that appears to the right.

After applying these rules, the original NL sentence becomes as follows:

`If` TEAM:(our) UNUM : (4) `has the ball ,`
TEAM : (our) UNUM : (4) `should` ACTION : (shoot) `.`

This transformation process results in the intermediate sequence that contains both NL words and MR productions. Such a sequence can be further processed by applying the following rules:

TEAM UNUM `has the ball`	$\Rightarrow$	CONDITION : *bowner*(TEAM UNUM)
TEAM UNUM `should` ACTION	$\Rightarrow$	DIRECTIVE : *do*(TEAM UNUM ACTION)

Note that in the above rules, the left hand side is a pattern that involves both NL words and semantic categories. Application of these two rules results in the following form:

`If` CONDITION : *bowner*(our 4) `,` DIRECTIVE : *do*(our 4 shoot) `.`

Finally, with one more rule:

`If` CONDITION `,` DIRECTIVE `.` $\Rightarrow$ RULE:(CONDITION DIRECTIVE)

the complete MR is obtained:

$$\text{RULE} : (bowner(\text{our } 4)\ do(\text{our } 4 \text{ shoot}))$$

In SILT, Kate et al. developed algorithms for learning such transformation rules as the above ones. The rules are learned from the corpus by constructing positive and negative examples. The learning algorithm is a heuristics-based approach that makes sure the learned rules are not overly general, such that the rules can cooperate with each other to generate the training corpus. On the other hand, the rules also need to be general enough so as not to cause over-fitting. The overall architecture of the SILT system is illustrated in Figure 2.3.

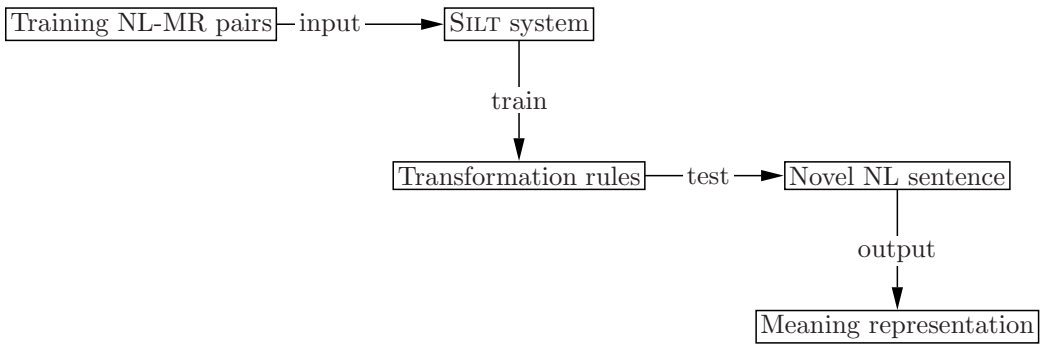


Figure 2.3: Architecture for the SILT system

2.2.2 The Wasp System

The WASP (Word Alignment-based Semantic Parsing) system [WM06] presents an algorithm for semantic parsing motivated by statistical machine translation techniques. The system considers the problem of transforming from natural language to meaning representations as a conventional machine translation task. First, WASP assumes that the MR can be represented with a formal context-free grammar. The system takes in pairs of natural language sentences and meaning representations, and learns transformation rules in the form of synchronous

context free grammar (SCFG) [LS68] that bridges natural language sentences and meaning representations. The basic assumption is that the NL-MR pair is constructed by a sequence of applications of the SCFG rules, as illustrated below:

$\langle \text{RULE}_1, \text{RULE}_1 \rangle$
$\Rightarrow$ $\langle$ `if` CONDITION_1 `,` DIRECTIVE_2 `.`,
$(\text{CONDITION}_1, \text{DIRECTIVE}_2)\rangle$
$\Rightarrow$ $\langle$ `if` TEAM_1 `player` UNUM_2 `has the ball ,` DIRECTIVE_3 `.`,
$(bowner(\text{TEAM}_1\ \text{UNUM}_2)\ \text{DIRECTIVE}_3)\rangle$
$\Rightarrow$ $\langle$ `if our player` UNUM_1 `has the ball ,` DIRECTIVE_2 `.`,
$(bowner(\text{our}\ \text{UNUM}_1)\ \text{DIRECTIVE}_2)\rangle$
$\Rightarrow$ $\langle$ `if our player 4 has the ball ,` DIRECTIVE_1 `.`,
$(bowner(\text{our}\ 4)\ \text{DIRECTIVE}_1)\rangle$
$\Rightarrow$ ...
$\Rightarrow$ $\langle$ `if our player 4 has the ball , then our player 6 should stay in the left side of our half .`,
$((bowner(\text{our}\ 4)\ do(\text{our}\ 6\ pos(left(half(\text{our}))))))\rangle$

where each step is an application of a single SCFG rule. For example, the SCFG rule associated with the second step of the above derivation process is as follows:

CONDITION
$\rightarrow$ $\langle \text{TEAM}_1$ `player` UNUM_2 `has the ball`, $bowner(\text{TEAM}_1\ \text{UNUM}_2)\rangle$

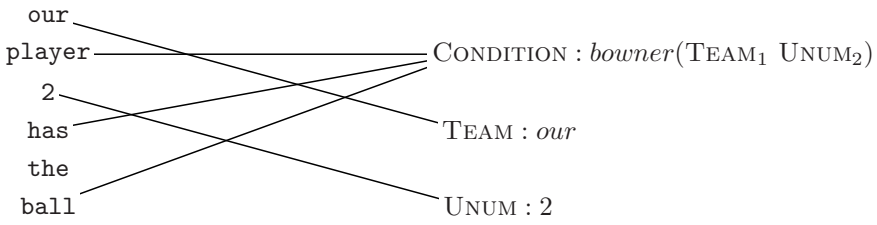


Figure 2.4: Example output from the word alignment model

Unlike SILT where syntactic labels are required for the tree-based transformation model, there is completely no syntactic information required by WASP. It assumes the MR can be linearized with a preorder traversal of the MR tree. The problem is next cast as a statistical machine translation problem, which is to translate from NL to the linearized MR. The system first takes as input pairs of NL sentences together with their corresponding MRs. Next the system tries to induce a set of SCFG rules from such a parallel corpus. The IBM word alignment model [BPPM93] was used to effectively find the associations between NL words and MR productions. An example partial alignment between NL words and MR productions is given in Figure 2.4. This forms the basis for the next step – extraction of the candidate SCFG transformation rules. Specifically, first the GIZA++ toolkit [ON03] was used to perform the word alignment task. Next, an algorithm was developed that makes use of certain heuristics to extract possible candidate SCFG rule entries. This phase is referred to as the lexicon acquisition phase as it proposes the set of all possible lexical entries for the translation task. After the lexicon acquisition phase is over, the system needs to learn the weight associated with each SCFG rule entry. However, the correct derivation associated with each instance is not explicitly given in the parallel corpus. Therefore, the correct derivation is treated as a hidden variable, and a log-linear model with expectation maximization (EM) [RKPJ00] is used to learn the weights associated with SCFG rules. Consider all possible derivations that yield the input NL sentence $\mathbf{e}$, the probability of a derivation $\mathbf{d}$ given $\mathbf{e}$ is as follows:

$$P_\lambda(\mathbf{d}|\mathbf{e}) = \frac{1}{Z_\lambda(\mathbf{e})} \exp \sum_i \lambda_i f_i(\mathbf{d}) \tag{2.1}$$

where the λ values are learned with the algorithm, and $Z_\lambda(\mathbf{e})$ is the normalization function. The MR produced by the most probable derivation is the output.

During the testing phase, given a novel NL sentence, the parser tries to find

the most probable derivation that yields the NL sentence. The corresponding MR associated with this derivation is returned as the output of the semantic parsing task. The detailed algorithm for WASP can be found in Wong's Ph.D thesis [Won07].

2.2.3 The Krisp System

The KRISP (Kernel-based Robust Interpretation for Semantic Parsing) [KM06] system makes use of support vector machines (SVM) [CST00] with string kernels [LSST+02] for semantic parsing.

The system considers MRs to be constructed compositionally from natural language sentences. The algorithm learns the semantic parser using an iterative approach, where at each iteration the parser is improved from the old one. At each iteration, the algorithm learns for each MR production a classifier. The classifier performs classification on natural language substrings and returns a probability score for each substring to correspond to that MR production. Positive and negative examples for each MR production are collected and refined at each iteration, and supplied to SVM for training the classifiers.

During the testing phase, with the help of the learned classifiers, the most probable semantic derivation for a novel NL sentence can be determined. In particular, an extended Earley algorithm [Ear70] was employed for this step. An example semantic derivation for the NL sentence "`which rivers run through the states bordering Texas ?`" is shown in Figure 2.5. Such a derivation exactly gives the corresponding MR for the NL sentence.

The probability associated with a derivation $\mathbf{d}$ is:

$$P(\mathbf{d}) = \prod_{\pi,[i..j]\in\mathbf{d}} P_{\pi}(s[i..j]) \tag{2.2}$$

The most probable partial derivation for the MR tree rooted by the production

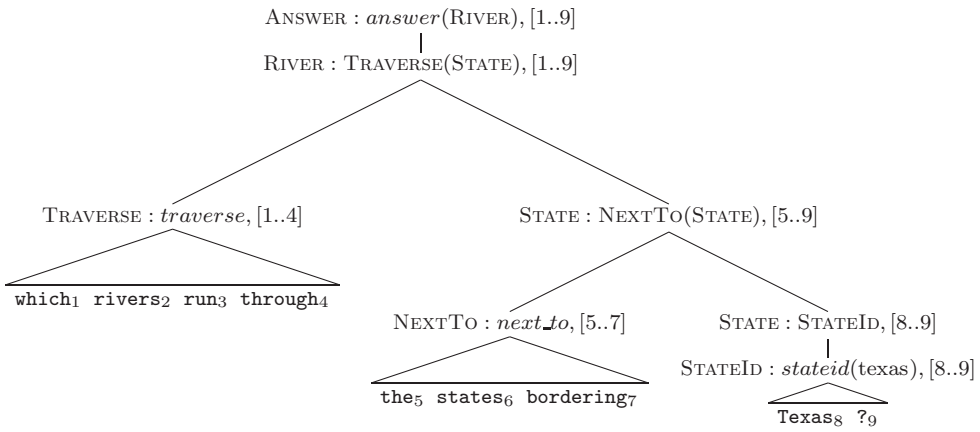


Figure 2.5: An example semantic derivation

n that covers the NL subsequence $s[i..j]$ is recursively defined as follows:

$$E^*_{n,s[i..j]} = \textsc{MakeTree}\left(\operatorname*{argmax}_{\pi,(p_1..p_t)}\left(P_\pi(s[i..j])\prod_{k=1..t} P(E^*_{n_k,p_k})\right)\right) \tag{2.3}$$

where $\pi = n \rightarrow n_1..n_t \in G$ and $(p_1..p_t) \in \textsc{Partition}(s[i..j], t)$. The function PARTITION returns the set of all partitions of $s[i..j]$ with t elements including their permutations.

There are also several related works built on top of this KRISP model. Specifically, SEMISUP-KRISP [KM07b] is an algorithm that allows the use of un-annotated sentences to improve the performance for semantic parsing. KRISPER [KM07a] is another algorithm that allows training over ambiguous supervision.

2.2.4 The Scissor System

The SCISSOR (Semantic Composition that Integrates Syntax and Semantics to get Optimal Representations) system [GM05] presents a model that generates a semantically augmented parse tree (SAPT). A semantically augmented parse tree is a tree whose nodes are labeled with both syntactic and semantic tags. Figure 2.6 gives an example SAPT.

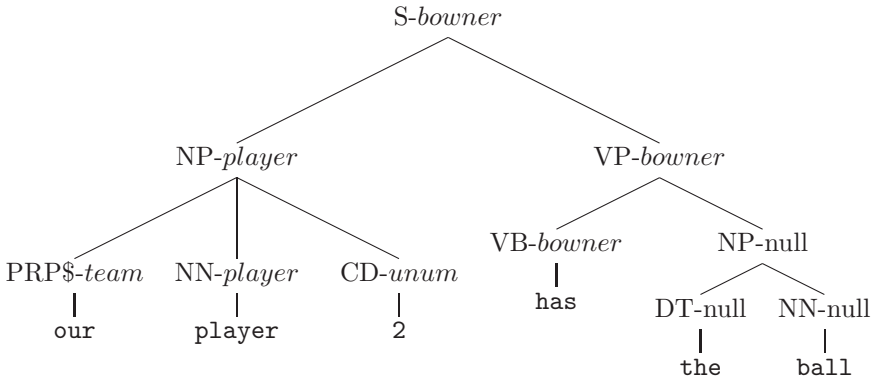


Figure 2.6: An example SAPT (semantically augmented parse tree)

In the training phase, an extended version of the Collins' syntactic parser [Col03] was used to train a parser for SAPTs. The learned parser is able to produce SAPTs which have two labels for internal tree nodes rather than one label as for syntactic parse trees. During testing, the learned parser can be applied to the new NL sentence, and produce a SAPT for it. The resulting SAPT contains the semantic labels which can be used to compose the MR. An example is shown in Figure 2.7.

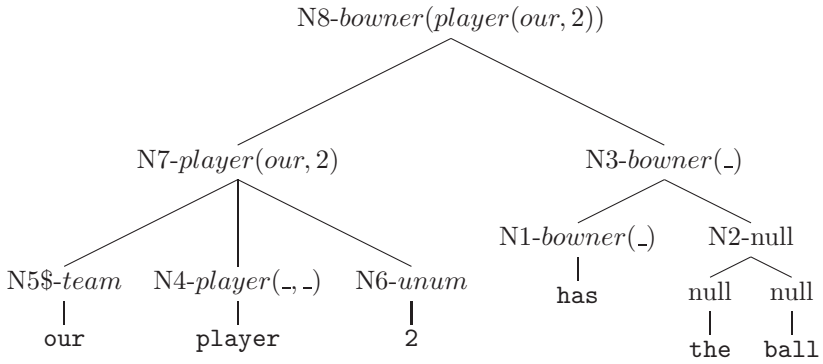


Figure 2.7: An example showing MR construction from the SAPT in Figure 2.6

An extension of the SCISSOR system named SCISSOR+ was also presented in [GM06]. This extended system was built on top of the original SCISSOR by augmenting it with an additional step for discriminative reranking [CK05]. Features from both syntactic and semantic labels are explored for reranking.

2.2.5 The Z&C Systems

Zettlemoyer and Collins presented systems that handle semantic parsing with lambda calculus [ZC05, ZC07]. Specifically, they make use of the Combinatory Categorial Grammar (CCG) [Ste96, Ste00] as the grammar formalism during the parsing process, mainly due to its advantages on a wide range of treatment of linguistic phenomena, and its convenience on incorporating both syntax and semantics during the parsing process.

An example NL sentence and its corresponding MR in logical form are as follows:

Sentence: `what states border texas`

Logical Form: $\lambda x.state(x) \wedge borders(x, texas)$

To parse the above NL-MR pair, a lexicon is required. Example lexical entries are as follows:

`Utah` := $NP : utah$

`Idaho` := $NP : idaho$

`borders` := $(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$

The algorithm presented in [ZC05] learns weighted CCG rules through a log-linear model. Specifically, define $\mathbf{f}(x, y)$ as a feature vector that represents a parse tree y paired with the NL sentence x. The algorithm learns a weight vector $\mathbf{w}$ through the training corpus. Thus, the optimal parse for the input sentence x given the lexicon Λ is defined as:

$$y^*(x) = \underset{y \in \mathrm{GEN}(x;\Lambda)}{\operatorname{argmax}} \mathbf{w} \cdot \mathbf{f}(x, y) \tag{2.4}$$

where the GEN function returns the list of all possible parses associated with the NL sentence x under the lexicon Λ. The goal of the algorithm is to learn the weight vector $\mathbf{w}$, as well as the appropriate lexicon Λ. An algorithm for jointly

learning both the weight vector and the lexicon is presented in [ZC05], which is further improved in [ZC07].

To learn the lexicon that contains a list of lexical entries, a set of rules are used to extract candidate lexical entries. Such rules are formally defined with a function called GENLEX that is applied to all the training instances. A partial list of rules contained by GENLEX is illustrated in Figure 2.8. Refer to [ZC05, ZC07] for a complete list of rules.

Rules		Example categories produced
Input Trigger	Output Category	
constant c	$NP : c$	$NP : boston$
arity one predicate p	$N : \lambda x.p(x)$	$N : \lambda x.flight(x)$
arity one predicate p	$S\backslash NP : \lambda x.p(x)$	$S\backslash NP : \lambda x.flight(x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(y,x)$	$(S\backslash NP)/NP : \lambda x.\lambda y.from(y,x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(x,y)$	$(S\backslash NP)/NP : \lambda x.\lambda y.from(x,y)$
arity one predicate p_1	$N/N : \lambda g.\lambda x.p_1(x) \wedge g(x)$	$N/N : \lambda g.\lambda x.flight(x) \wedge g(x)$
literal with arity two predicate p_2 and constant second argument c	$N/N : \lambda g.\lambda x.p_2(x,c) \wedge g(x)$	$N/N : \lambda g.\lambda x.from(x,boston) \wedge g(x)$
arity one function f	$S/NP : \lambda x.f(x)$	$S/NP : \lambda x.cost(x)$

Figure 2.8: A partial list of rules used in GENLEX. Example categories are produced based on the logical form: argmax($\lambda x.flight(x) \wedge from(x, boston), \lambda x.cost(x)$)

Given the learned lexical entries, the parse tree that results in the given NL-MR pair is illustrated in Figure 2.9. In CCG, there exists a set of combinators that are used to combine categories to form larger structures with both syntax and semantics.

$$\dfrac{\dfrac{\dfrac{\texttt{Utah}}{NP} \quad \dfrac{\texttt{borders}}{(S\backslash NP)/NP} \quad \dfrac{\texttt{Idaho}}{NP}}{\dfrac{utah \quad \lambda x.\lambda y.border(y,x) \quad idaho}{(S\backslash NP)}\ >}}{\dfrac{\lambda y.borders(y, idaho)}{S}\ <}$$

$$borders(utah, idaho)$$

Figure 2.9: An example of CCG parses

Specifically, in the paper of [ZC05], only the combinators *functional application* rules were considered. In the extended system [ZC07], additional com-

binators such as the *relaxed functional application* rules and *crossed functional composition* rules were exploited.

2.3 Natural Language Generation

The natural language generation task has a long history in the NLP field. Some earlier systems that generate natural language texts from underlying meaning representation include, for example, the SURGE system [ER96], the KPML system [Bat96], and the RealPro system [LR97]. However, these systems rely on an underlying manually created linguistic knowledge base for processing. In contrast, Nitrogen [LK98] and the NLG system by Ratnaparkhi [Rat00] use corpus-based statistical approaches which are trainable and therefore more robust. Nitrogen assumes the meaning representations are of flexible graphical structure that involves both syntactic and semantic information. The system by Ratnaparkhi assumes that the meaning representations are in the form of attribute-value pairs.

In the next two subsections, we specifically discuss the recent systems introduced by Wong and Mooney [WM07a], where the meaning representation is in the functional form as discussed in Section 2.1.

2.3.1 The Pharaoh and Pharaoh++ System

The Pharaoh [Koe04] system is a recent phrase-based statistical machine translation (SMT) system. Wong and Mooney [WM07a] explored the method of utilizing Pharaoh for generating natural language from meaning representations. We first give an overview of the Pharaoh system.

The Pharaoh system, as a phrase-based machine translation system, first requires a bilingual lexicon that consists of pairs of phrases in both languages. Such a lexicon is obtained by collecting consistent alignments between source and target languages with the Giza++ toolkit.

Next, the following log-linear model defines the conditional probability for each candidate translation **e** given the input sentence **f**.

$$\begin{aligned}\Pr_\lambda(\mathbf{e}|\mathbf{f}) \quad \propto \quad & \Pr_{LM}(\mathbf{e})^{\lambda_1} \prod_{i=1}^{I} \Big(P(\bar{f}_i|\bar{e}_i)^{\lambda_2} P(\bar{e}_i|\bar{f}_i)^{\lambda_3} P_w(\bar{f}_i|\bar{e}_i)^{\lambda_4} P_w(\bar{e}_i|\bar{f}_i)^{\lambda_5} \\ & d(i-1,i)^{\lambda_6} \exp(-|\bar{e}_i|)^{\lambda_7} \exp(-1)^{\lambda_8} \Big) \end{aligned} \tag{2.5}$$

In the above formula, $\Pr_{LM}(\mathbf{e})$ is the language model. The symbols $\bar{e}$ and $\bar{f}$ denotes the phrases extracted from target and source sentences respectively. The symbol I denotes the number of such phrase pairs. The terms $P(\bar{e}|\bar{f})$ and $P(\bar{f}|\bar{e})$ denotes the conditional probabilities which are estimated with relative frequencies of $\bar{e}$ and $\bar{f}$. The terms $P_w(\bar{e}|\bar{f})$ and $P_w(\bar{f}|\bar{e})$ are the lexical weights as described in [KOM03]. The distortion term $d(i,j)$ gives the cost for reordering phrases at position i and j. The remaining two terms $\exp(-|e|)$ and $\exp(-1)$ are word penalty term and sentence penalty term respectively, which give some control over the output sentence length. The parameters λ are trained using the minimum error rate training algorithm as introduced by Och [Och03]. For more details, refer to [KOM03].

Wong and Mooney [WM07a] considered the problem of transforming meaning representations to natural language sentences as a machine translation problem. They first presented a basic system that makes use of no MR structural information. Next they introduced an improved system named PHARAOH++ over the basic system, which exploits MR information.

In the basic system PHARAOH as presented in their paper [WM07a], they simply treat the MR as a sequence of tokens, which forms the source language sentence. The target language is the output NL sentence. For example, the source language sentence can be as follows:

$$answer(state(traverse\_1(riverid(`ohio'))))$$

The target sentence is the NL sentence "`what states does the Ohio run through ?`".

In the improved system PHARAOH++ , the input source MR is no longer a stream of symbols. Instead, MR productions are considered, and the linearized MR tree is considered the input to the word alignment model.

2.3.2 The Wasp^{-1} and Wasp^{-1}++ System

The WASP^{-1} system is a system inverted from the WASP semantic parsing system discussed in section 2.2.2.

The most probable natural language sentence that corresponds to the given MR tree $\mathbf{f}$ is defined as follows:

$$\mathbf{e}^* = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e})\Pr(\mathbf{f}|\mathbf{e}) \tag{2.6}$$

Equation 2.6 is based on the noisy-channel model, where the term $\Pr(\mathbf{e})$ is the language model, and the term $\Pr(\mathbf{f}|\mathbf{e})$ is the translation model.

With the Viterbi approximation, the most probable sentence can be expressed as follows:

$$\begin{aligned} \mathbf{e}^* &= \max_{\mathbf{e}} \Pr(\mathbf{e})\Pr(\mathbf{f}|\mathbf{e}) \\ &\approx \max_{\mathbf{d} \in D(G|\mathbf{f})} \Pr(\mathbf{e}(\mathbf{d}))\Pr_\lambda(\mathbf{d}|\mathbf{e}(\mathbf{d})) \\ &= \max_{\mathbf{d} \in D(G|\mathbf{f})} \frac{\Pr(\mathbf{e}(\mathbf{d})) \cdot \exp \sum_i \lambda_i f_i(\mathbf{d})}{Z_\lambda(\mathbf{e}(\mathbf{d}))} \end{aligned} \tag{2.7}$$

where the function $D(G|\mathbf{f})$ returns all the possible derivations under the SCFG G, given the input $\mathbf{f}$. The language model probability can be computed with an n-gram model, while the translation model probability score is computed by re-using the log-linear model of WASP, as given in Equation 2.1.

The WASP^{-1}++ system is an improved model over WASP by borrowing the models from PHARAOH.

$$\Pr_\lambda(\mathbf{d}|\mathbf{f}) \propto \Pr(\mathbf{e}(\mathbf{d}))^{\lambda_1} \prod_{d \in \mathbf{d}} w_\lambda(r(d)) \tag{2.8}$$

where $\prod_{d \in \mathbf{d}} w_\lambda(r(d))$ is the product of the weights of the SCFG rules used in a derivation $\mathbf{d}$. The weight w_λ is in turn defined as follows:

$$w_\lambda(A \rightarrow \langle \alpha, \beta \rangle) = P(\beta|\alpha)^{\lambda_2} P(\alpha|\beta)^{\lambda_3} P_w(\beta|\alpha)^{\lambda_4} P_w(\alpha|\beta)^{\lambda_5} \exp(-|\alpha|)^{\lambda_6} \tag{2.9}$$

where Ps are the relative frequencies, and P_ws are the lexical weights. The only penalty term $\exp(-|\alpha|)$ gives some control over output sentence length.

Now the most probable output sentence $\mathbf{e}^*$ is defined as follows:

$$\mathbf{e}^* = \mathbf{e}\left(\underset{\mathbf{d} \in D(G|\mathbf{f})}{\operatorname{argmax}} \Pr_\lambda(\mathbf{d}|\mathbf{f})\right) \tag{2.10}$$

In this WASP^{-1}++ system, several additional techniques are also introduced. First, different from WASP and WASP^{-1}, where unaligned NL words are usually replaced with word gaps, WASP^{-1}++ eliminates word gaps. This is because empirically, using known words usually leads to better fluency in NLG task. Second, extracted short rules are merged to form longer ones.

As pointed out by Wong [Won07], the main difference between PHARAOH++ and WASP^{-1}++ is that the former only allows contiguous lexical entries, while the latter allows discontiguous lexical entries.

Chapter 3

The Hybrid Tree Framework

In this chapter, we present our proposed *hybrid tree* framework. We first introduce several notation in Section 3.1, followed by some discussions that lead to the framework in Section 3.2.

3.1 Notation

For ease of discussion, we formally define the following terms and introduce the notation as follows:

- A ***word*** is a basic unit of natural language, which is usually defined over a *vocabulary* $\{w_1, \ldots, w_V\}$.

  We use w to denote a single word.

- A ***word sequence*** is a contiguous sequence of words.

  We use $\mathbf{w}$ to denote a word sequence.

- A ***sentence*** is a word sequence that has complete semantics.

  We use $\widetilde{\mathbf{w}}$ to denote a sentence.

- We use $\mathcal{M}$ to denote a semantic category;

- We use m_x to denote a single MR production;

  Sometimes we also use the expanded version $\mathcal{M}_x : p_\chi(\pi)$ to denote a single MR production, where $\mathcal{M}_x$ is the semantic category for this production, p_χ is the function symbol, and π is the argument list. The argument list can consist of child semantic categories or constants. For example, in the MR production $\mathcal{M}_x : p_\chi(\mathcal{M}_y\ \mathcal{M}_z)$, $\mathcal{M}_y$ and $\mathcal{M}_z$ are the child semantic categories;

- We use $\widehat{m_x}$ to denote an MR tree rooted by the MR production m_x.

  In particular, we also use $\widetilde{\mathbf{m}}$ to denote an MR tree that corresponds to a complete NL sentence;

- We use $\widehat{m_x}\downarrow$ to denote child MR subtrees below the MR production m_x in the MR tree $\widehat{m_x}$. For example, for the MR tree shown in Figure 3.1, $\widehat{m_a}\downarrow \equiv \langle \widehat{m_b}, \widehat{m_c} \rangle$, and $\widehat{m_b}\downarrow \equiv \langle \widehat{m_d} \rangle$.

- An ***N-M pair*** is a contiguous NL word sequence paired with its corresponding MR tree.

  We use d to denote an N-M pair.

- A ***corpus*** consists of a set of N-M pairs.

  Each N-M pair appearing in the corpus is associated with a unique index. We use d^i to denote the N-M pair with index i, and use $\mathbf{d}$ to denote the corpus.

3.2 The Framework

Our framework is heavily motivated by the **principle of compositionality**. The most general form of the principle can be expressed as follows:

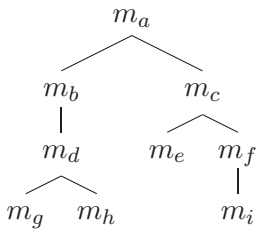


Figure 3.1: An example MR tree

Principle of Compositionality (Frege's Principle) The meaning of an expression is a function of the meanings of its parts and of the way they are syntactically combined. [Par82]

The principle of compositionality states that the meaning of a complex natural language expression is determined by the meanings of the constituents, and the way these constituents are organized. However, the definitions of "meaning" and "syntax" are missing from this principle.

Since in this thesis we identify the meanings with MR trees, we attempt to link the principle of compositionality with the notion of MR trees. Each MR tree should represent some form of semantics which should be unambiguous. Naturally, we can make the simple assumption that the semantics conveyed by an MR tree $\widehat{m_x}$ is fully determined by the root MR production m_x and its child MR subtrees $\widehat{m_x}\downarrow$. Let us denote the semantics of an MR tree $\widehat{m_x}$ as $\mathcal{S}(\widehat{m_x})$, the semantics of an MR production m_x as $\mathcal{S}(m_x)$, and the semantics of the MR subtrees $\widehat{m_x}\downarrow$ as $\mathcal{S}(\widehat{m_x}\downarrow)$. Based on the above discussion, we make the following assumption.

Assumption 3.1.

$$\forall \widehat{m_x}, \exists g \text{ such that } \mathcal{S}(\widehat{m_x}) \equiv g\Big(\mathcal{S}(m_x), \mathcal{S}(\widehat{m_x}\downarrow)\Big) \tag{3.1}$$

This assumption states that for any MR tree $\widehat{m_x}$, the semantics of this MR tree can be equivalently represented with a function of the semantics of the MR production m_x and the semantics of the subtrees $\widehat{m_x}\downarrow$.

Now the issue turns to how to select the function g. Note that the above formula does not involve any natural language. Naturally, we can believe that $\mathcal{S}(m_x)$, which is the semantics of m_x, can be correspond with NL sequences selected from a vocabulary.

Based on this belief, we make the following assumption.

Assumption 3.2.

$$\forall \widehat{m_x}, \exists g', \exists \Phi \text{ such that } \mathcal{S}(\widehat{m_x}) \equiv g'\Big(\Phi(m_x, \boldsymbol{V}), \mathcal{S}(\widehat{m_x}\downarrow)\Big) \tag{3.2}$$

The function Φ selects from the vocabulary $\mathbf{V}$ several NL word sequences that correspond to the semantics of m_x. The function g' is defined over $\Phi(m_x, \mathbf{V})$ and $\mathcal{S}(\widehat{m_x}\downarrow)$, such that $\widehat{m_x}$ is semantically equivalent to the semantics returned by the function g'. In other words, the function g' is defined over NL word sequences as well as semantics of child MR subtrees, which tells how the NL words and MR productions are organized together. This is in fact a strong assumption where we assume that semantics can be represented with natural language words in an unambiguous manner. Though in practice this assumption is not always true, we believe this is quite a reasonable approach for building the relationship between NL and MR.

Note that the above formula is defined in a recursive manner. In other words, the semantics of subtrees $\widehat{m_x}\downarrow$ can again be rewritten as functions of a similar form. We therefore can repeatedly replace each MR production appearing in the MR tree with NL words. We call such a process *semantic expansion*.

Therefore the input arguments of the function g' is a sequence that contains NL words interleaved with MR subtrees. One important fact is that, if we keep applying the semantic expansion process, we obtain a list of NL word sequences which can form a complete NL sentence. This sentence corresponds exactly to the semantics of the complete MR tree $\widehat{m_x}$.

Let us formally introduce several concepts.

Definition 3.1. *A **hybrid sequence** consists of NL word sequences intermixed with MR productions.*

For example, $\overline{\mathbf{w}_1\ m_a\ \mathbf{w}_2\ m_b\ \mathbf{w}_3}$ is a hybrid sequence, where $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ are NL word sequences, while m_a and m_b are two MR productions.

Definition 3.2. *An **abstract hybrid sequence** consists of NL word sequences intermixed with semantic categories.*

For example, $\overline{\mathbf{w}_1\ \mathcal{M}_a\ \mathbf{w}_2\ \mathcal{M}_b}$ is an abstract hybrid sequence, where $\mathbf{w}_1$, $\mathbf{w}_2$ are NL word sequences, while $\mathcal{M}_a$ and $\mathcal{M}_b$ are two semantic categories.

With these definitions, the above semantic expansion process in fact repeatedly creates hybrid sequences below each MR production while traversing the MR tree. The complete structure produced by the semantic expansion results in a hybrid tree, as defined below.

Definition 3.3. *A **hybrid tree** is a tree consisting of NL words as leaves and MR productions as internal nodes.*

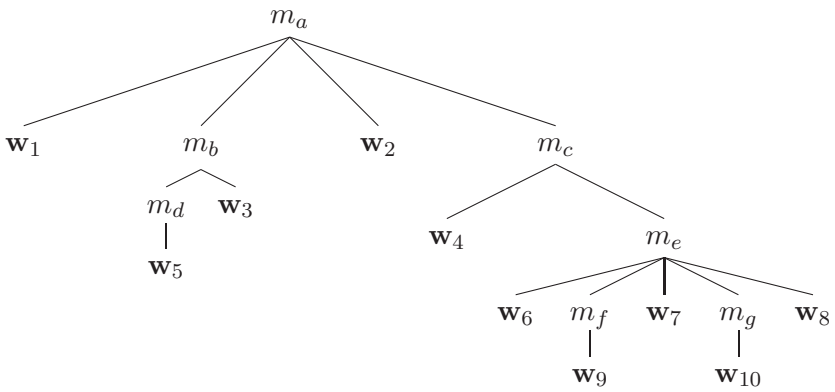


Figure 3.2: An example hybrid tree

An example hybrid tree is shown in Figure 3.2. The symbols $\mathbf{w}_1, \ldots, \mathbf{w}_{10}$ are NL word sequences, and $m_a, \ldots, m_g$ are MR productions. Note that each level of the hybrid tree is exactly a hybrid sequence.

A hybrid tree can be constructed by performing some manipulations on the MR tree. Typically, it can be constructed from an MR tree by inserting NL words as leaves, and reordering child MR productions below a particular MR production. With the concept of hybrid tree, we further introduce some constraints on the functions Φ and g' mentioned in Assumption 3.2. Specifically, for an MR tree $\widehat{m_x}$, the function $\Phi(\widehat{m_x}, \mathbf{V})$ returns several NL word sequences, *e.g.*, $\mathbf{w}_1$ and $\mathbf{w}_2$. These NL word sequences together with child MR productions $\widehat{m_x} \downarrow$ form a particular hybrid sequence below the MR production m_x, *e.g.*, $\overline{\mathbf{w}_1\ m_y\ \mathbf{w}_2\ m_z}$. Repeating such a process, we eventually build a hybrid tree. Thus, the function Φ selects the NL word sequences that corresponds to the MR production m_x, and the function g' determines how the NL word sequences and the child MR productions of m_x are interleaved.

This leads to the following important assumption which forms the basis of our proposed hybrid tree framework. This assumption is also referred to as the **hybrid tree assumption**:

Assumption 3.3. (Hybrid Tree Assumption) *For each N-M pair $\langle \widehat{m_x}, \boldsymbol{w} \rangle$, there exists a hybrid tree $\mathcal{T}$, such that $\mathcal{T}$ contains $\widehat{m_x}$, and the yield of $\mathcal{T}$ is exactly $\boldsymbol{w}$.*

The models and algorithms presented in the rest of the thesis are based on the above basic assumption.

Chapter 4

The Generative Models for Hybrid Tree

In this chapter, we discuss models built on top of the proposed hybrid trees in the framework introduced in the previous section. We first present the generative models built on top of the framework in Section 4.1, followed by the methods for parameter estimation in Section 4.2.

4.1 The Generative Process

The hybrid tree assumption states that each N-M pair can be contained by a hybrid tree. However, such a hybrid tree is not explicitly given. It is natural to believe that there exists an underlying generative model, from which such an N-M pair can be generated.

Let us take the hybrid tree given in Figure 3.2 for example. We can assume that such a hybrid tree is generated from an underlying generative model.

Figure 4.1 shows the generative process. The model first selects a semantic category $\mathcal{M}_a$, from which an MR production m_a is generated. Given the MR production m_a, the model next selects an abstract hybrid sequence $\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}$,

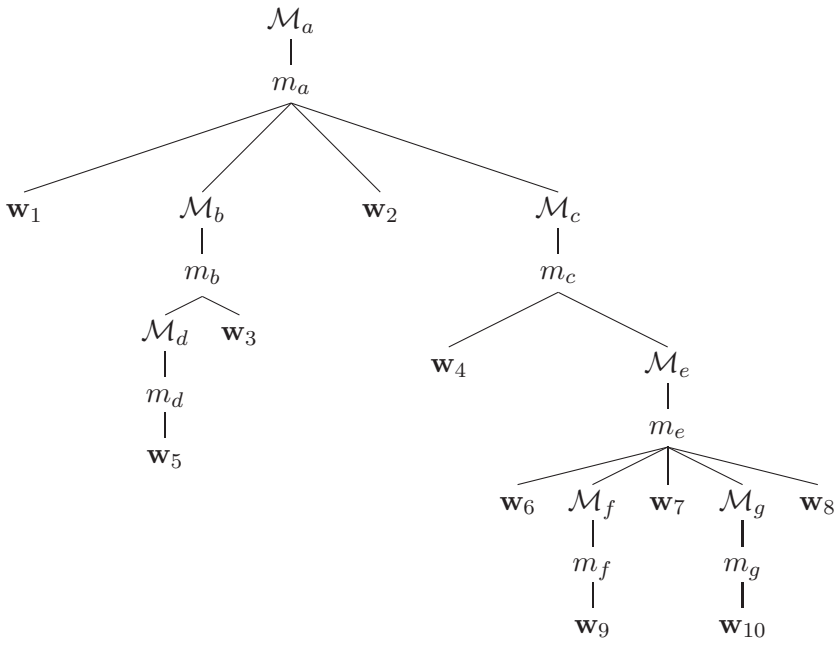


Figure 4.1: The generative process

which consists of NL words as well as m_a's child semantic categories $\mathcal{M}_b$ and $\mathcal{M}_c$.

Next, two child MR productions m_b and m_c are generated. These two productions will in turn generate other hybrid sequences and productions, recursively. This process produces a hybrid tree $\mathcal{T}$, whose nodes are either NL words or MR productions. Given this tree, we can recover an NL sentence $\mathbf{w}$ as the yield of the hybrid tree. We can recover the MR tree $\widehat{m_a}$ from the hybrid tree as well.

With several conditional independence assumptions, the probability of generating $\langle \widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T} \rangle$ is defined as:

$$\begin{aligned} P(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T}) &= P(\mathcal{M}_a) \times P(m_a|\mathcal{M}_a) \times P(\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}|m_a) \\ &\times P(m_b|m_a, \arg_1) \times P(\overline{\mathcal{M}_d\ \mathbf{w}_3}|m_b) \\ &\times P(m_d|m_b, \arg_1) \times P(\overline{\mathbf{w}_5}|m_d) \\ &\times P(m_c|m_a, \arg_2) \times P(\overline{\mathbf{w}_4\ \mathcal{M}_e}|m_c) \\ &\times P(m_e|m_c \arg_1) \times P(\overline{\mathbf{w}_6\ \mathcal{M}_f\ \mathbf{w}_7\ \mathcal{M}_g\ \mathbf{w}_8}|m_e) \\ &\times P(m_f|m_e, \arg_1) \times P(\overline{\mathbf{w}_9}|m_f) \\ &\times P(m_g|m_e, \arg_2) \times P(\overline{\mathbf{w}_{10}}|m_g) \end{aligned} \tag{4.1}$$

where "arg_k" refers to the position of the child semantic category (k-th child semantic category) in the argument list.

Figure 4.2 gives an example hybrid tree for the training example from Section 2.1.

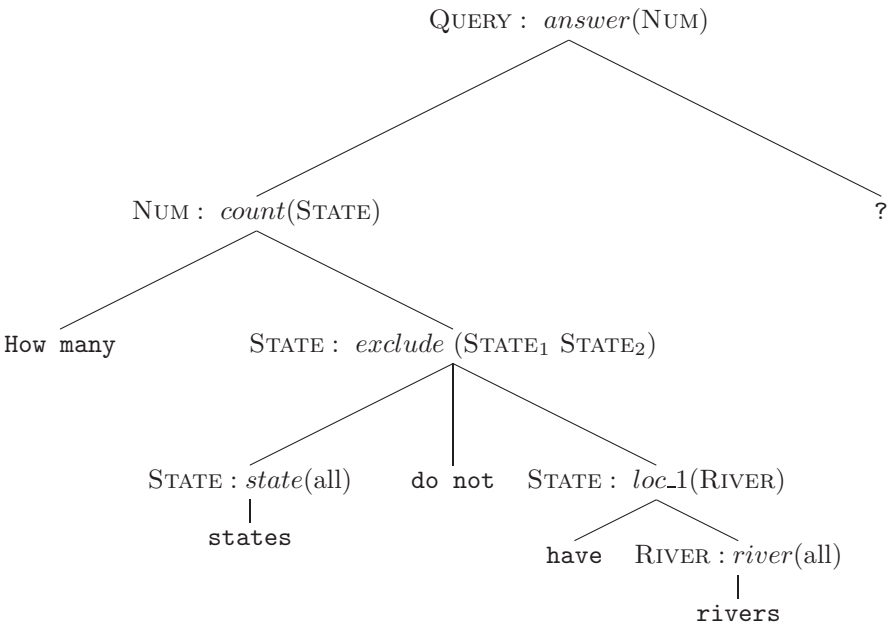


Figure 4.2: An example hybrid tree for the GEOQUERY corpus

4.1.1 Decomposition with Hybrid Patterns

We notice that Equation 4.1 not only models the generation of MR productions, but also models the generation of hybrid sequences from MR productions. However, direct modeling of the generation process for hybrid sequences will suffer from data sparseness. We therefore seek alternative parameterization for the generative model.

Inspired by Collins' syntactic parsing models [Col03], we consider the generation process for a hybrid sequence from an MR production as a Markov process.

Firstly let us introduce the notion of hybrid pattern, as follows.

Definition 4.1. *For a given MR production, a* ***hybrid pattern*** *consists of NL*

word sequences intermixed with the position indices of the child MR productions.

For example, assume in an MR tree $\widehat{m_a}$, the MR production m_a has only 1 child MR production m_b. The possible hybrid sequences below m_a in any hybrid tree could be one of the following: 1) Just the first MR production alone; 2) An NL word sequence followed by the first MR production; 3) The first MR production followed by an NL word sequence; 4) An NL word sequence, the first MR production, and then followed by another NL word sequence.

In principle, an MR production can have an arbitrary number of child semantic categories. For simplicity, we assume that each MR production has at most two child semantic categories in its argument list. This is a reasonable assumption, since any production can be transformed into a sequence of productions of this form. Given this assumption, we can list in Table 4.1 all possible hybrid patterns.

If we denote the MR production as m, the first child MR production as $\mathcal{Y}$, and the second as $\mathcal{Z}$, we obtain the following possible hybrid patterns, for the case of one child MR production: $m \rightarrow \mathcal{Y}$, $m \rightarrow \mathbf{w}\mathcal{Y}$, $m \rightarrow \mathcal{Y}\mathbf{w}$, and $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}$.

A complete list of hybrid patterns for MR productions with different number of child MR productions is shown in Table 4.1.

In this table, m is an MR production, $\mathcal{Y}$ and $\mathcal{Z}$ are respectively the first and second child semantic category in m's argument list. The symbol $\mathbf{w}$ refers to a contiguous sequence of NL words. The last 8 rows contain hybrid patterns that reflect reordering of one production's child semantic categories during the generation process. For example, consider the case that the MR production RULE : (CONDITION$_1$, DIRECTIVE$_2$) generates the following hybrid sequence:

$$\overline{\texttt{do}\ \text{DIRECTIVE}_2\ \texttt{,}\ \texttt{if}\ \text{CONDITION}_1}$$

, the hybrid pattern $m \rightarrow \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y}$ is associated with this generation step. Note

Number of child MRs	Hybrid pattern	Number of patterns
0	$m \rightarrow \mathbf{w}$	1
1	$m \rightarrow \mathcal{Y}$	4
	$m \rightarrow \mathbf{w}\mathcal{Y}$	
	$m \rightarrow \mathcal{Y}\mathbf{w}$	
	$m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}$	
2	$m \rightarrow \mathcal{Y}\mathcal{Z}$	8
	$m \rightarrow \mathbf{w}\mathcal{Y}\mathcal{Z}$	
	$m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z}$	
	$m \rightarrow \mathcal{Y}\mathcal{Z}\mathbf{w}$	
	$m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}$	
	$m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}$	
	$m \rightarrow \mathbf{w}\mathcal{Y}\mathcal{Z}\mathbf{w}$	
	$m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}$	
	$m \rightarrow \mathcal{Z}\mathcal{Y}$	8
	$m \rightarrow \mathbf{w}\mathcal{Z}\mathcal{Y}$	
	$m \rightarrow \mathcal{Z}\mathbf{w}\mathcal{Y}$	
	$m \rightarrow \mathcal{Z}\mathcal{Y}\mathbf{w}$	
	$m \rightarrow \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y}$	
	$m \rightarrow \mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w}$	
	$m \rightarrow \mathbf{w}\mathcal{Z}\mathcal{Y}\mathbf{w}$	
	$m \rightarrow \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w}$	

Table 4.1: The complete list of hybrid patterns

that in this example, there is a reordering between the two child semantic categories.

For the example hybrid tree in Figure 3.2, we can decompose the probability for generating the hybrid sequence as follows:

$$\begin{aligned} P(\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}|m_a) &= P(m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}|m_a) \times P(\mathbf{w}_1|m_a) \\ &\times P(\mathcal{M}_b|m_a, \mathbf{w}_1) \times P(\mathbf{w}_2|m_a, \mathbf{w}_1, \mathcal{M}_b) \\ &\times P(\mathcal{M}_c|m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2) \times P(\texttt{END}|m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2, \mathcal{M}_c) \end{aligned} \quad (4.2)$$

Note that unigram, bigram, or trigram assumptions can be made here for generating NL words and semantic categories. For example, under a bigram assumption, the second to last term can be written as $P(\mathcal{M}_c|m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2) \equiv P(\mathcal{M}_c|m_a, w_2^k)$, where w_2^k is the last word in $\mathbf{w}_2$. We call such additional infor-

mation that we condition on, the ***context***.

The joint probability for the hybrid tree in Figure 4.2 is as follows:

$$
\begin{aligned}
P(\mathbf{w}, \mathbf{m}, \mathcal{T}) \;=\;& P(\textsc{Query} : \mathit{answer}(\textsc{Num}) | -, \arg 1) \\
\times\;& P(\textsc{Num}_1 \texttt{ ?} | \textsc{Query} : \mathit{answer}(\textsc{Num})) \\
\times\;& P(\textsc{Num} : \mathit{count}(\textsc{State}) | \textsc{Query} : \mathit{answer}(\textsc{Num}), \arg 1) \\
\times\;& P(\texttt{How many } \textsc{State}_1 | \textsc{Num} : \mathit{count}(\textsc{State})) \\
\times\;& P(\textsc{State} : \mathit{exclude}(\textsc{State}_1\ \textsc{State}_2) | \textsc{Num} : \mathit{count}(\mathit{State}), \arg 1) \\
\times\;& P(\textsc{State}_1 \texttt{ do not } \textsc{State}_2 | \textsc{State} : \mathit{exclude}(\textsc{State}_1\ \textsc{State}_2)) \\
\times\;& P(\textsc{State} : \mathit{state}(\text{all}) | \textsc{State} : \mathit{exclude}(\textsc{State}_1\ \textsc{State}_2), \arg 1) \\
\times\;& P(\texttt{states} | \textsc{State} : \mathit{state}(\text{all})) \\
\times\;& P(\textsc{State} : \mathit{loc\_1}(\textsc{River}) | \textsc{State} : \mathit{exclude}(\textsc{State}_1\ \textsc{State}_2), \arg 2) \\
\times\;& P(\texttt{have } \textsc{River}_1 | \textsc{State} : \mathit{loc\_1}(\textsc{River})) \\
\times\;& P(\textsc{River} : \mathit{river}(\text{all}) | \textsc{State} : \mathit{loc\_1}(\textsc{River}), \arg 1) \\
\times\;& P(\texttt{rivers} | \textsc{River} : \mathit{river}(\text{all})) \qquad (4.3)
\end{aligned}
$$

which consists of terms responsible for generating MR as well as terms responsible for generating hybrid sequences.

The latter terms can be further decomposed. An example is given as follows:

$$
\begin{aligned}
P(\texttt{How many } \textsc{State}_1 | \textsc{Num} : \mathit{count}(\textsc{State})) \;=\;& P(m \rightarrow \mathbf{w}\mathcal{Y} | \textsc{Num} : \mathit{count}(\textsc{State})) \\
\times\;& P(\texttt{How} | \textsc{Num} : \mathit{count}(\textsc{State}), \texttt{BEGIN}) \\
\times\;& P(\texttt{many} | \textsc{Num} : \mathit{count}(\textsc{State}), \texttt{How}) \\
\times\;& P(\textsc{State}_1 | \textsc{Num} : \mathit{count}(\textsc{State}), \texttt{many}) \\
\times\;& P(\texttt{END} | \textsc{Num} : \mathit{count}(\textsc{State}), \textsc{State}) \\
& \qquad (4.4)
\end{aligned}
$$

which consists of a term responsible for selecting hybrid patterns, as well as terms responsible for generating NL words and semantic categories. We make the bigram assumption here, as to be discussed in next section.

4.2 Parameter Estimation

The corpus consists of pairs of NL sentences and MR trees. Our task is to learn a hybrid tree for each N-M pair. Once the hybrid tree is learned, we can make use of

the learned model parameters to perform some useful tasks such as transforming novel NL sentences to their corresponding MR trees, or transforming novel MR trees to their NL sentences.

There are three categories of parameters used in the model, as summarized below.

1. **MR model parameters**. These parameters model the generation of new MR productions from their parent MR productions, *e.g.,* $P(m_b|m_a, \text{arg}_1)$.

   We denote them with $\rho(m'|m_j, \text{arg}_k)$. Note that these parameters satisfy the following constraints:

   $$\forall j. \sum_{m'} \rho(m'|m_j, \text{arg}_k) = 1 \tag{4.5}$$

   where $k = 1, 2$ if m_j has two child semantic categories, and $k = 1$ if m_j has only one child semantic category.

2. **Emission parameters**. These parameters model the generation of a hybrid sequence from an MR production, *e.g.,* $P(\mathbf{w}_1|m_a)$, $P(\mathcal{M}_b|m_a, \mathbf{w}_1)$.

   We denote them with $\theta(t|m_j, \Lambda)$. Note that these parameters satisfy the following constraints:

   $$\forall j. \sum_{t} \theta(t|m_j, \Lambda) = 1 \tag{4.6}$$

   where t is an NL word, the "`END`" symbol, or a semantic category. Λ is the context associated with m_j and t.

3. **Pattern parameters.** These parameters model the selection of a hybrid pattern given an MR production, *e.g.,* $P(m \rightarrow \mathbf{w}\mathcal{Y}|m_a)$.

   We denote them with $\phi(r|m_j)$. Note that these parameters satisfy the following constraints:

$$\forall j. \sum_{r} \phi(r|m_j) = 1 \tag{4.7}$$

where r is a hybrid pattern listed in Table 4.1.

Note that the context Λ for the emission parameters is not yet explicitly given. In fact, with different context assumptions, we reach different variations of the model. In particular, we consider three assumptions, as follows:

Unigram Model We make the following assumption:

$$\theta(t_k|m_j, \Lambda) = P(t_k|m_j) \tag{4.8}$$

where t_k is a semantic category or an NL word, and m_j is an MR production.

In other words, the generation of the next NL word depends on its direct parent MR production only.

Bigram Model We make the following assumption:

$$\theta(t_k|m_j, \Lambda) = P(t_k|m_j, t_{k-1}) \tag{4.9}$$

where t_{k-1} is the semantic category or NL word to the left of t_k, *i.e.,* the previous semantic category or NL word.

In other words, the generation of the next NL word depends on its direct parent MR production as well as the previously generated NL word or semantic category only.

Mixgram Model We make the following assumption:

$$\theta(t_k|m_j, \Lambda) = \frac{1}{2} \times \Big[P(t_k|m_j) + P(t_k|m_j, t_{k-1})\Big] \tag{4.10}$$

This model can be viewed as an interpolation of unigram and bigram models.

Next we present algorithms for parameter estimation for the three categories of parameters respectively.

4.2.1 Modeling Meaning Representation

The MR model parameters can be estimated independently from the other two. In fact, no matter how the actual hybrid tree looks, it always contains the same MR tree as given, and the generation of the next child MR production with a certain child production index purely depends on the parent MR production. These parameters can be analogously viewed as the "language model" parameters for the MR tree, and can be estimated directly from the corpus by simply reading off the counts of occurrences of MR productions in MR tree over the training corpus.

The maximum-likelihood estimation gives us:

$$\hat{\rho}(m_i|m_j, \arg_k)_{\text{MLE}} = \frac{\text{Count}(m_i \text{ appeared as the } k\text{-th child of } m_j \text{ in the corpus})}{\text{Count}(m_j \text{ appeared in the corpus})} \tag{4.11}$$

This formula has an intuitive meaning. The probability of choosing the next MR production m_i below another MR production m_j at child position k is equal to the number of occurrences of m_i that appears as the k-th child of the MR production m_j over the total number of occurrences of m_j in the corpus. A formal derivation of this formula is provided in Appendix A.

In practice, to resolve data sparseness problem, a variant of the bigram Katz Back-Off Model [Kat87] is employed here for smoothing. The formulas are as follows:

- Define two sets:

$$\mathcal{A}(m_j, k) = \{m : \text{Count}(m_j, k, m) > 0\}$$
$$\mathcal{B}(m_j, k) = \{m : \text{Count}(m_j, k, m) = 0\}$$

where $\text{Count}(m_j, k, m)$ denotes the number of times the MR production m appears as the k-th child of the MR production m_j.

The set $\mathcal{A}(m_j, k)$ contains all such MR productions that appear as the k-th child of the MR production m_j, and $\mathcal{B}(m_j, k)$ is the set of MR productions that do not appear as the k-th child of the MR production m_j.

- The smoothed model is then defined as follows:

  1. If $m_i \in \mathcal{A}(m_j, k)$

$$\hat{\rho}(m_i|m_j, \arg = k)_{\text{KATZ}} = \frac{\text{Count}(m_j, k, m_i) - 0.5}{\text{Count}(m_j)} \tag{4.12}$$

  2. If $m_i \in \mathcal{B}(m_j, k)$

$$\hat{\rho}(m_i|m_j, \arg = k)_{\text{KATZ}} = \Delta(m_j, k) \times \frac{\hat{\rho}_{\text{MLE}}(m_i|m_j, \arg = k)}{\sum_{m \in \mathcal{B}(m_j,k)} \hat{\rho}_{\text{MLE}}(m|m_j, \arg = k)} \tag{4.13}$$

  where

$$\Delta(m_j, k) = 1 - \sum_{m \in \mathcal{A}(m_j,k)} \frac{\text{Count}(m_j, k, m) - 0.5}{Count(m_j)}$$

Note that this category of parameters only needs to be learned from the training corpus once only.

4.2.2 Learning the Generative Parameters

Learning the remaining two categories of parameters is more challenging. In a conventional PCFG parsing task [Col03], during the training phase, the correct correspondence between NL words and syntactic structures is fully accessible.

In other words, there is a single deterministic derivation associated with each training instance. Therefore model parameters can be directly estimated from the training corpus by counting. However, in our task, the correct correspondence between NL words and MR trees is unknown. Many possible derivations could reach the same N-M pair, where each such derivation forms a hybrid tree.

This problem is analogous to the scenario in PCFG parsing, where the correct syntactic parses for the sentences are missing, but the set of parsing rules is provided. Typically, such a problem is solved by the Expectation-Maximization (EM) algorithm [DLR77], where the correct parses for sentences are constructed with hidden variables. The algorithm designed for PCFG parsing, in contrast to the forward-backward algorithm for HMM, is usually referred to as the inside-outside algorithm [Bak79].

Similarly, for this specific task, the hybrid tree is constructed using hidden variables and estimated from the training set. An efficient inside-outside style algorithm can be used for model estimation, similar to that used in [YK01], as discussed next.

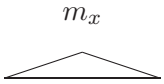
The Inside-Outside Style Algorithm

In this section, we discuss how to estimate the emission and pattern parameters with the Expectation Maximization (EM) algorithm, by using an inside-outside style dynamic programming approach.

Before we move to the detailed algorithm, we firstly formally introduce the following notations.

- We use $d^i \equiv \left\langle \widetilde{\mathbf{m}}^i, \widetilde{\mathbf{w}}^i \right\rangle$ to denote the i-th N-M pair that appears in the corpus, consisting of the MR tree $\widetilde{\mathbf{m}}^i$ and the NL sentence $\widetilde{\mathbf{w}}^i$.

  We also use $d_v \equiv \langle \widehat{m_v}, \mathbf{w}_v \rangle$ to denote an N-M pair, where the MR subtree rooted by MR production m_v will correspond to (*i.e.,* hierarchically generate) the NL word sequence $\mathbf{w}_v$.

- We use  to denote an N-M pair, where the NL word sequence is $w_1 \ \ldots \ w_n$ and the MR subtree is $\widehat{m_x}$.

For example, let us consider the N-M pair as shown in Figure 4.3. The correct hybrid tree for this N-M pair is not explicitly given. In principle, there exists many possible hybrid trees for such a given N-M pair. However, it is not practical to enumerate all of them, and therefore we seek alternative compact ways to represent all the hybrid trees.

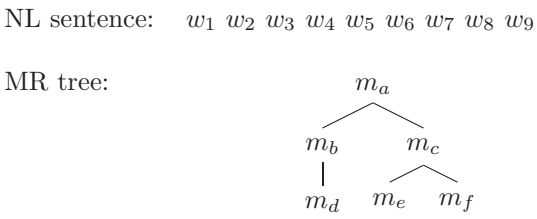
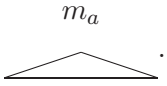


Figure 4.3: An example N-M pair

Following the notation introduced above, the N-M pair in Figure 4.3 can also be expressed with  . In fact, this N-M pair implicitly represents all possible hybrid trees for this NL sentence - MR tree pair. Let us look at the following four possible hybrid trees as shown in Figure 4.4.

Note that these four hybrid trees share one important common property: the hybrid sequences directly below m_a for each of them are identical, which is $\overline{m_b \ w_4 \ w_5 \ m_c}$. Since the generative model encodes some independence assumptions at each level, this motivates us to introduce some compact representation for hybrid trees that share common hybrid sequences at a certain level. Such a compact representation may help us to easily perform parameter estimation.

We first introduce the notion of decomposed N-M pair.

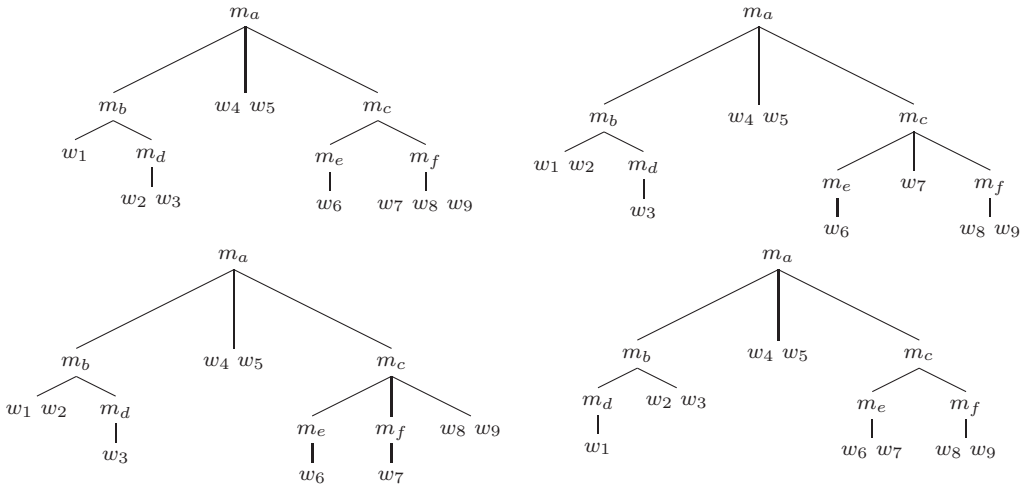
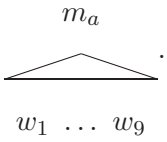


Figure 4.4: Four possible hybrid trees

Definition 4.2. *A **decomposed N-M pair** is a tree whose root is an MR production and the children are N-M pairs.*

A decomposed N-M pair can be regarded as an N-M pair whose top level has been expanded to a hybrid sequence. Figure 4.5 gives two example decomposed N-M pairs for the N-M pair 

.

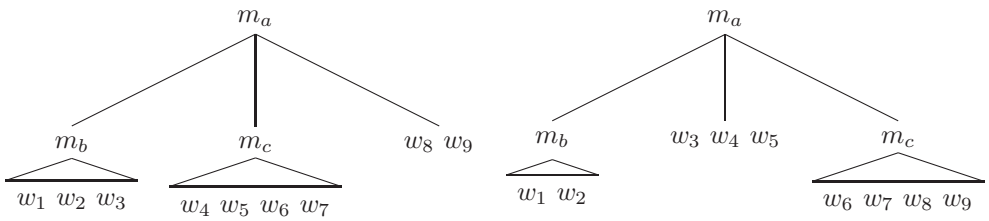


Figure 4.5: Two example decomposed N-M pairs

Given the above definition, we can conclude that Figure 4.6 gives a compact representation for all the hybrid trees that have the hybrid sequence $\overline{m_b\ w_4\ w_5\ m_c}$ appearing below m_a. Thus, all the four hybrid trees in Figure 4.4 are included by the decomposed N-M pair as shown in Figure 4.6.

Notice that each decomposed N-M pair in turn contains other N-M pairs as

its children, which can be further rewritten as decomposed N-M pairs. Thus, we can build a graph where each node is either an N-M pair or a decomposed N-M pair, and the edges model the parent-child relationships. In this way, with the notion of decomposed N-M pair, we successfully find a way to represent hybrid trees in a compact manner.

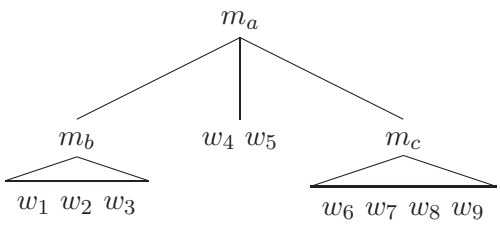


Figure 4.6: The compact representation - a decomposed N-M pair

The graph that models the relationship between N-M pairs and decomposed N-M pairs is shown in Figure 4.7, where we use h to denote decomposed N-M pairs and d to denote N-M pairs. Each N-M pair can be decomposed into a set of possible decomposed N-M pairs (*i.e.*, a disjunctive node), while each decomposed N-M pair contains 0, 1, or 2 N-M pairs as children (*i.e.*, a conjunctive node). We call the graph an ***N-M pairs graph***.

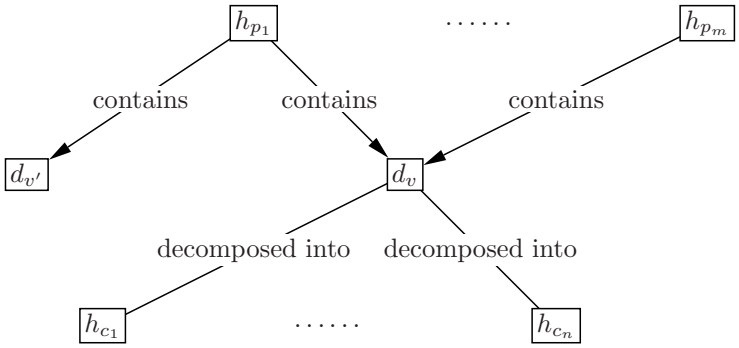


Figure 4.7: The N-M pairs graph representing the relation between N-M pairs and decomposed N-M pairs.

Given a tree structure, we can easily establish certain relationships between the nodes in the tree. To describe the relationships better, we formally introduce

several definitions as follows.

Definition 4.3. *The function* CHILDREN(d_v) *returns the set of all possible decomposed N-M pairs d_v can be decomposed into.*

For example, the function CHILDREN $\left(\begin{array}{c} m_c \\ \triangle \\ w_6\ w_7\ w_8\ w_9 \end{array} \right)$ consists of the list of possible decomposed N-M pairs as shown in Figure 4.8.

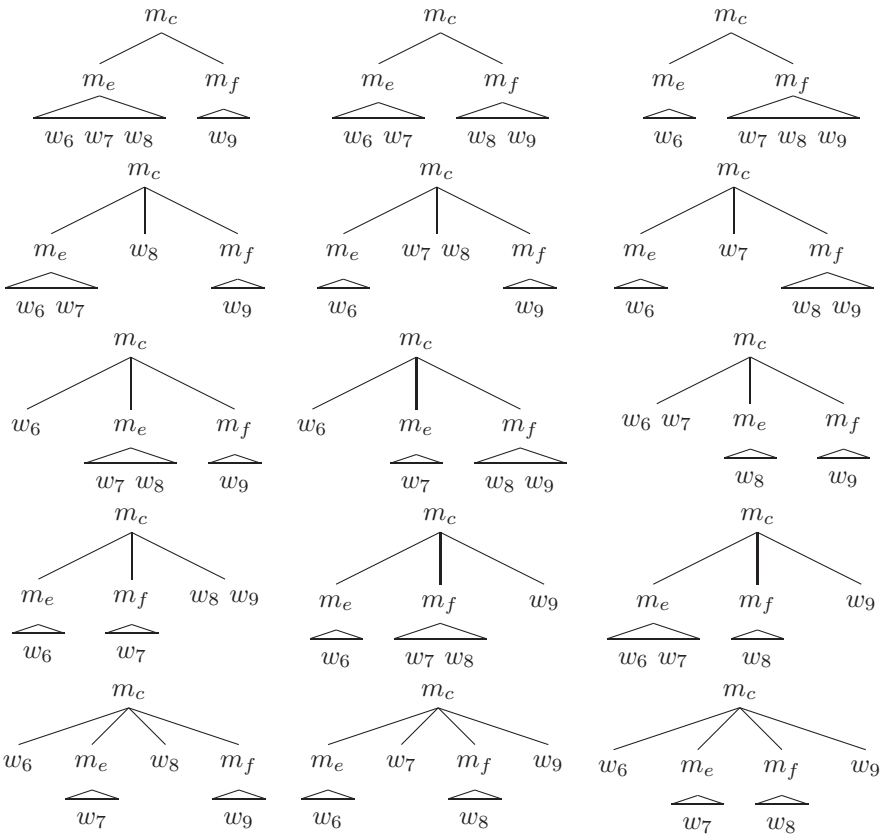


Figure 4.8: All possible decomposed N-M pairs

Definition 4.4. *The function* PARENT(d_v) *returns the set of all possible decomposed N-M pairs under which the N-M pair d_v can be generated.*

For example, the function PARENT $\left(\begin{array}{c} m_c \\ \triangle \\ w_6\ w_7\ w_8\ w_9 \end{array} \right)$ consists of the list of possible hybrid trees as shown in Figure 4.9.

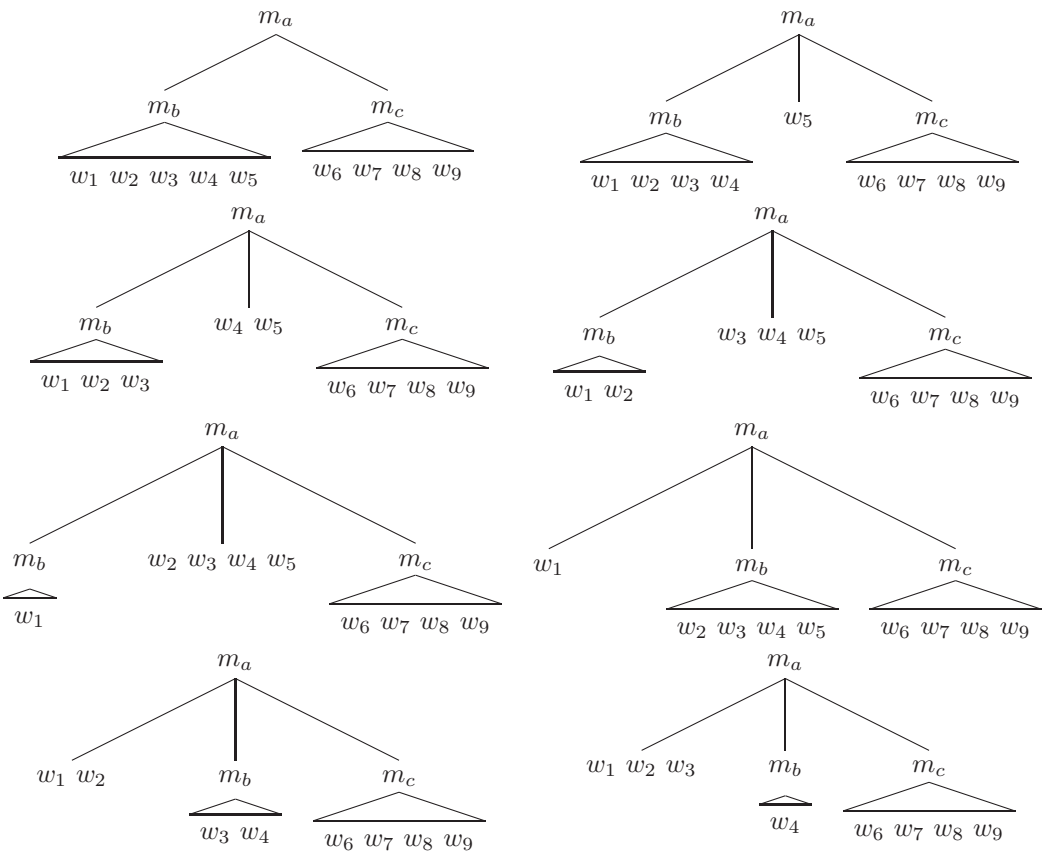


Figure 4.9: All possible hybrid trees

Definition 4.5. *The function* PARENT(h) *gives the unique N-M pair that can be decomposed as* h*.*

For example,

$$\textsc{Parent}\left(\begin{array}{c} m_c \\ m_e \quad\quad m_f \\ w_6\ w_7\ w_8 \quad w_9 \end{array}\right) = \begin{array}{c} m_c \\ w_6\ w_7\ w_8\ w_9 \end{array} \tag{4.14}$$

Definition 4.6. *The function* $\textsc{Children}(h)$ *gives the list of N-M pairs that appear directly below the decomposed N-M pair* h*.*

For example,

$$\textsc{Children}\left(\begin{array}{c} m_c \\ m_e \quad\quad m_f \\ w_6\ w_7\ w_8 \quad w_9 \end{array}\right) = \left\langle \begin{array}{c} m_e \\ w_6\ w_7\ w_8 \end{array}, \begin{array}{c} m_f \\ w_9 \end{array} \right\rangle \tag{4.15}$$

Figure 4.7 gives a graph representing the relations between the entities.

Given the above definitions, we can make use of a CKY-style parse chart for tracking the probabilities. The formulas for computing inside and outside probabilities as well as the equations for updating parameters are given as follows. For a complete derivation of these formulas, refer to Appendix A.

For each training N-M pair instance, we first build its corresponding N-M pairs graph, and then compute the inside and outside probabilities according to the following formulas.

The inside (β) probabilities The inside probabilities are defined as follows.

1. If $d_v \equiv \langle \widehat{m_v}, \mathbf{w}_v \rangle$ is a leaf node in the N-M pairs graph,

$$\beta(d_v) = P(\mathbf{w}_v | m_v) \tag{4.16}$$

2. If $d_v \equiv \langle \widehat{m_v}, \mathbf{w}_v \rangle$ is not a leaf node in the N-M pairs graph,

$$\beta(d_v) = \sum_{h \in \textsc{Children}(d_v)} \left(P(h|\mathbf{m}_v) \times \prod_{d_{v'} \in \textsc{Children}(h)} \beta(d_{v'}) \right) \quad (4.17)$$

Specifically, we define $\beta(d^i)$ as the inside probability for the i-th N-M instance pair.

The outside (α) probabilities The outside probabilities are defined as follows.

1. If $d_v \equiv \langle \widehat{m_v}, \mathbf{w}_v \rangle$ is the root node in the N-M pairs graph.

$$\alpha(d_v) = 1 \quad (4.18)$$

2. If $d_v \equiv \langle \widehat{m_v}, \mathbf{w}_v \rangle$ is not the root node in the N-M pairs graph.

$$\alpha(d_v) = \sum_{h \in \textsc{Parent}(d_v)} \left(\alpha\big(\textsc{Parent}(h)\big) \times P\big(h|\textsc{Parent}(h)\big) \times \right. \\ \left. \times \prod_{d_{v'} \in \textsc{Children}(h), v' \neq v} \beta(d_{v'}) \right) \quad (4.19)$$

Parameter Update The equations for parameter update are as follows.

1. Updating the emission parameter.

   We define the count $c^i(t, m_v, \Lambda_k; \Omega)$ based on the old parameter Ω, where t is either an NL word or a semantic category, for an instance N-M pair $d^i \equiv \langle \widetilde{\mathbf{m}}^i, \widetilde{\mathbf{w}}^i \rangle$, as follows:

$$c^i(t, m_v, \Lambda_k; \Omega) = \frac{1}{\beta(d^i)} \times \sum_{\substack{(t, m_v, \Lambda_k) \in h \\ h \in \textsc{Children}(m_v)}} \left(\alpha(d^i_v) \times \right. \\ \left. \times P(h|m_v) \times \prod_{d^i_{v'} \in \textsc{Children}(h)} \beta(d^i_{v'}) \right) \quad (4.20)$$

where $(t, m_v, \Lambda_k) \in h$ means the tuple (t, m_v, Λ_k) appears in the decomposed N-M pair h.

The emission parameter is re-estimated as:

$$\theta'(t|m_v, \Lambda_k) = \frac{\sum_i c^i(t, m_v, \Lambda_k; \Omega)}{\sum_{t'} \sum_i c^i(t', m_v, \Lambda_k; \Omega)} \tag{4.21}$$

2. Updating the pattern parameter

We define the count $c^i(r, m_v; \Omega)$ based on the old parameter Ω, where r is a hybrid pattern, for an instance N-M pair $d^i \equiv \langle \widetilde{\mathbf{m}}^i, \widetilde{\mathbf{w}}^i \rangle$:

$$c^i(r, m_v; \Omega) = \frac{1}{\beta(d^i)} \times \sum_{\substack{(r, m_v) \in h \\ h \in \textsc{Children}(m_v)}} \left(\alpha(d^i_v) \times \right.$$
$$\left. \times P(h|m_v) \times \prod_{d^i_{v'} \in \textsc{Children}(h)} \beta(d^i_{v'}) \right) \tag{4.22}$$

where $(r, m_v) \in h$ means the tuple (r, m_v) appears in the decomposed N-M pair h.

The pattern parameter is re-estimated as:

$$\phi'(r|m_v) = \frac{\sum_i c^i(r, m_v; \Omega)}{\sum_{r'} \sum_i c^i(r', m_v; \Omega)} \tag{4.23}$$

The algorithm described here is an Expectation Maximization (EM) algorithm [DLR77]. The algorithm generally iterates over two sub-routines, namely the E (Expectation)-step and M (Maximization)-step. The E-step of the algorithm finds the expected number of occurrences for each possible configuration of our interest. In this case, this step consists of computation of the inside and outside scores, as well as computation of the counts (c values). The M-step of the algorithm updates the model's parameters, based on the expected counts that we have computed in E-step. In other words, Equations 4.21 and 4.23 are for the M-step.

Smoothing

It is reasonable to believe that different MR productions that share identical function symbols are likely to generate NL words with similar distribution, regardless of semantic categories. For example, RIVER : *largest* (RIVER) and CITY : *largest* (CITY) are both likely to generate the word "`biggest`".

In view of this, a smoothing technique can be deployed. We can assume half of the time words can be generated from the production's function symbol alone if it is not empty. Mathematically, assuming m_a with function symbol p_a, for a NL word or semantic category t, we have:

$$\theta(t|m_a,\Lambda) = \begin{cases} \theta_e(t|m_a,\Lambda) & \text{If } p_a \text{ is empty} \\ \big(\theta_e(t|m_a,\Lambda) + \theta_e(t|p_a,\Lambda)\big)/2 & \text{otherwise} \end{cases} \tag{4.24}$$

where θ_e models the generation of t from an MR production or its function symbol, together with the context Λ.

4.3 Adding Prior Knowledge

We also allow our system to take as input a minimal set of prior knowledge associated with each corpus. This set of prior knowledge may contain several facts specific to the corpus. However, learning such facts from the corpus requires substantial training instances, and users may wish to supply such knowledge directly to the system.

For example, the NL phrase

`baton rouge`

should always map to the following MR production:

CITYNAME : *baton_rouge*

One nice property of our model is that such prior knowledge can be easily incorporated into the system by creating them as several ***prior instances***. In other words, the above N-M pair will form a training instance and be put together with other training instances during parameter estimation routine. However, since such prior knowledge is believed to be true with stronger confidence, to reflect the significance of these instances, a larger weight ω is assigned to these prior instances, as if these instances have appeared ω times beforehand.

Mathematically, we would like to optimize the following objective function:

$$\mathcal{O} = \sum_i \log \Big(P(d^i) \Big)^{c_i} = \sum_i c_i \cdot \log P(\widetilde{m}^i, \widetilde{\mathbf{w}}^i) \tag{4.25}$$

where $(\widetilde{m}^i, \widetilde{\mathbf{w}}^i)$ is the i-th training instance N-M pair, and c_i refers to the number of times the i-th instance N-M pair appears in the training corpus. Thus, the weights for the original instances are all 1's, while the weights for the prior instances are all set to ω (in all our experiments performed in this thesis, we set ω to 1000).

Chapter 5

Efficient Algorithms for the Generative Models

Though the inside-outside approach already employs packed representations for dynamic programming, a naive implementation of the inference algorithm will still require $O(n^6m)$ time for 1 EM iteration, where n is the number of NL words in the sentence, and m is the number of MR productions in the MR tree. This is very expensive and sometimes not practical, as n and m can be potentially very large. Thus, efficient algorithms for training and decoding need to be developed, as to be discussed in this section.

5.1 Introduction

In this section, we develop an efficient dynamic programming algorithm that enables the inference to run in $O(n^3m)$ time. For clarity purpose, we primarily make the bigram assumption throughout our discussions in this section. The algorithm is also applicable to the model with unigram assumption, in which case the algorithm is in fact even simpler.

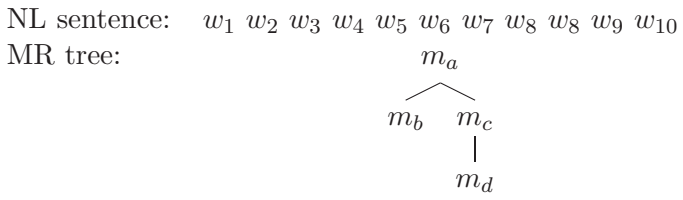


Figure 5.1: An example N-M pair

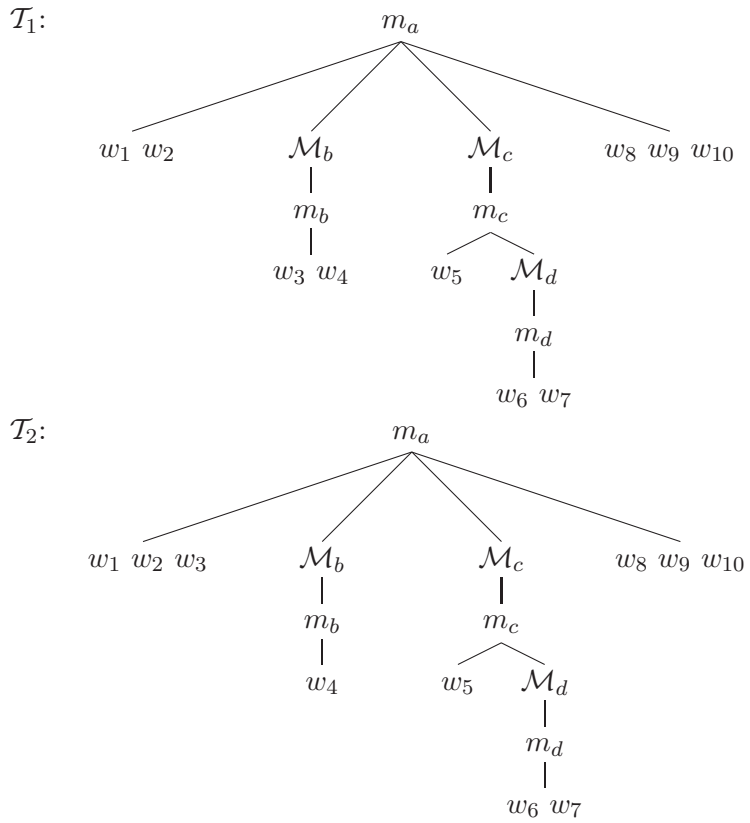


Figure 5.2: Two slightly different decomposed N-M pairs

To illustrate why a dynamic programming approach works for the models, it is best to start with an example. Figure 5.2 shows two possible hybrid trees, given the N-M pair in Figure 5.1, which form two different partial hybrid trees with slightly different hybrid sequences generated under the MR production m_a. We denote the two hybrid trees as $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively.

The joint probability scores associated with generating these two hybrid trees

are given in Equation 5.1 and Equation 5.2 respectively. As we can observe, the two joint probability formulas contain some common probability terms. This is not hard to understand, because the two hybrid trees share some common sub-structures, as illustrated in Figure 5.3. Note that in the figure, $m_x(\Lambda)$ refers to the partial sub-tree is rooted by the MR production m_x, while the context associated with generation of the next NL word or semantic category under m_x is Λ, with a bigram assumption. The identical sub-tree enclosed by the two rectangles is the important common sub-structure of the two hybrid trees $\mathcal{T}_1$ and $\mathcal{T}_2$.

In fact, Figure 5.3 also provides us an alternative view of the two hybrid trees. Specifically, if there is a methodology that computes the inside probability score for the sub-tree inside the rectangle, we only need to handle such computation once only over different hybrid trees which contain such a sub-tree. In this way, intuitively, we can avoid a large amount of redundant computations.

We found that there exist sophisticated algorithms that make good use of the independence assumptions required by the generative model, and enables the computation process done in an efficient manner. More details of the algorithms are to be presented in subsequent sections of this chapter.

$$
\begin{aligned}
P(\mathcal{T}_1) &= \phi(m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}|m_a) \times \theta(w_1|m_a, \texttt{BEGIN}) \times \theta(w_2|m_a, w_1) \\
&\times \theta(\mathcal{M}_b|m_a, w_2) \times \theta(\mathcal{M}_c|m_a, \mathcal{M}_b) \times \theta(w_8|m_a, \mathcal{M}_c) \\
&\times \theta(w_9|m_a, w_8) \times \theta(w_{10}|m_a, w_9) \times \theta(\texttt{END}|m_a, w_n) \\
&\times \rho(m_b|m_a, \arg_1) \times \phi(m \rightarrow \mathcal{Y}|m_a) \\
&\times \theta(w_3|m_b, \texttt{BEGIN}) \times \theta(w_4|m_b, w_3) \times \theta(\texttt{END}|m_b, w_4) \\
&\times \rho(m_c|m_a, \arg_2) \times \phi(m \rightarrow \mathbf{w}\mathcal{Y}|m_c) \\
&\times \theta(w_5|m_c, \texttt{BEGIN}) \times \theta(\mathcal{M}_d|m_c, w_5) \times \theta(\texttt{END}|m_c, \mathcal{M}_d) \\
&\times \rho(m_d|m_c, \arg_1) \times \phi(m \rightarrow \mathcal{Y}|m_d) \\
&\times \theta(w_6|m_d, \texttt{BEGIN}) \times \theta(w_7|m_d, w_6) \times \theta(\texttt{END}|m_d, w_7) \qquad (5.1)
\end{aligned}
$$

$$
\begin{aligned}
P(\mathcal{T}_2) &= \phi(m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}|m_a) \times \theta(w_1|m_a, \texttt{BEGIN}) \times \theta(w_2|m_a, w_1) \\
&\times \theta(w_3|m_a, w_2) \times \theta(\mathcal{M}_b|m_a, w_3) \times \theta(\mathcal{M}_c|m_a, \mathcal{M}_b) \times \theta(w_8|m_a, \mathcal{M}_c) \\
&\times \theta(w_9|m_a, w_8) \times \theta(w_{10}|m_a, w_9) \times \theta(\texttt{END}|m_a, w_n) \\
&\times \rho(m_b|m_a, \arg_1) \times \phi(m \rightarrow \mathcal{Y}|m_a) \\
&\times \theta(w_4|m_b, \texttt{BEGIN}) \times \theta(\texttt{END}|m_b, w_4) \\
&\times \rho(m_c|m_a, \arg_2) \times \phi(m \rightarrow \mathbf{w}\mathcal{Y}|m_c) \\
&\times \theta(w_5|m_c, \texttt{BEGIN}) \times \theta(\mathcal{M}_d|m_c, w_5) \times \theta(\texttt{END}|m_c, \mathcal{M}_d) \\
&\times \rho(m_d|m_c, \arg_1) \times \phi(m \rightarrow \mathcal{Y}|m_d) \\
&\times \theta(w_6|m_d, \texttt{BEGIN}) \times \theta(w_7|m_d, w_6) \times \theta(\texttt{END}|m_d, w_7) \qquad (5.2)
\end{aligned}
$$

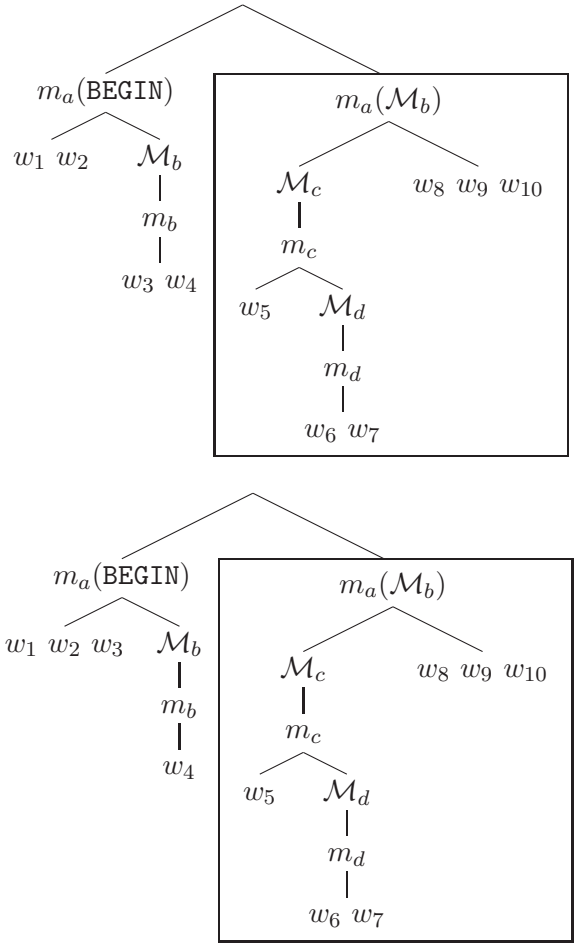


Figure 5.3: Alternative views for the two hybrid trees in Figure 5.2

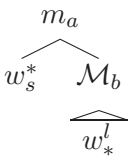
5.2 The Dynamic Programming Algorithms

5.2.1 Notation

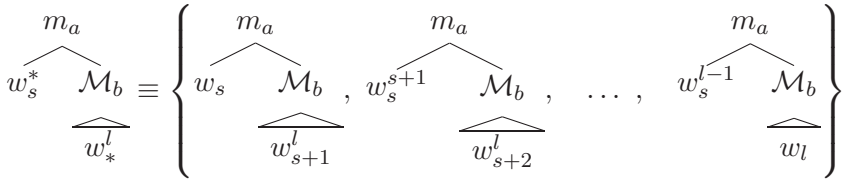
For the ease of discussion, we firstly introduce some useful notation.

- $w+$: an NL word sequence that consists of one or more NL words;
- w_s^*: an NL word sequence that starts with the NL word w_s;
- w_*^l: an NL word sequence that ends with the NL word w_l;
- w_s^l: an NL word sequence that starts with the NL word w_s and ends with the NL word w_l.

With the above notation, we can represent a set of hybrid trees that span the word sequence w_s^l and share identical hybrid patterns with a more compact representation. For example, the structure



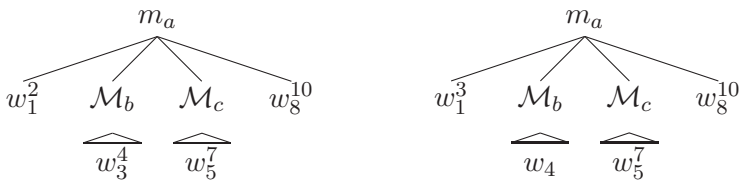
represents the set of hybrid trees each has the hybrid pattern $m \rightarrow \mathbf{w}\mathcal{Y}$ below the MR production m_a, and spans the word sequence w_s^l. Specifically, we have the following relation:



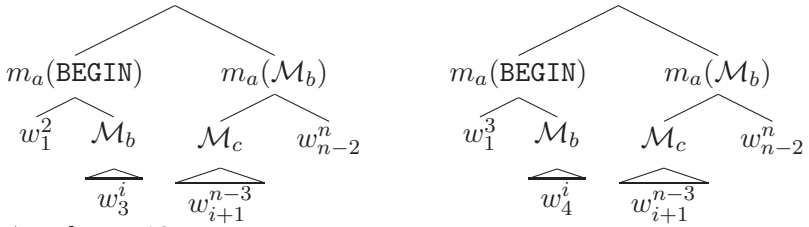
(5.3)

Such a compact representation actually consists of many hybrid trees that share similar hybrid patterns at a certain level of the trees. We therefore call such a structural representation an ***aggregated hybrid tree***.

The two hybrid trees in Figure 5.2 belong to the following two aggregated hybrid trees, respectively:



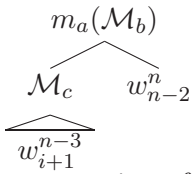
Given the independence assumption introduced by the generative model, the generation of the branch rooted by $\mathcal{M}_c$ as well as the NL word sequence w_8^{10} from the parent m_a depends on the MR production m_a and $\mathcal{M}_b$ only. We can therefore rewrite the above two representations as follows:



where $i = 4$ and $n = 10$.

In other words, the original aggregated hybrid tree can be rewritten as a conjunction of two aggregated hybrid trees. Note that in the new aggregated hybrid trees, the root MR production is associated with a context Λ, which is the preceding NL word or semantic category under the bigram assumption.

Note that these two representations share a common sub-tree, which is



It is not hard to observe that computation of the inside probabilities for the two aggregated hybrid trees both involve the computation of the inside probability for the above sub-tree. Due to the independence assumptions made by the generative model, the inside probability for this sub-tree needs to be computed once only.

5.2.2 Computing Inside Probabilities

We use $\mathcal{P}_\beta$ to denote the inside probability for a certain decomposed N-M pair. The standard procedure for computing the inside probabilities is given as Algo-

rithm 1.

Data: Training set $\widetilde{\mathbf{w}}^d, \widetilde{\mathbf{m}}^d$ for $d = 1, \ldots, n$
Result: The inside probabilities for each N-M pair
$L = |\widetilde{\mathbf{m}}^d|$;
for $l \leftarrow$ *0 to* $L - 1$ **do**
  for $i \leftarrow$ *1 to* $L - l$ **do**
    for m_x *as the next MR production in* $\widetilde{\boldsymbol{m}}^d$ *with post-order traversal order of* $\widetilde{\boldsymbol{m}}^d$ **do**
      if m_x *has single child* m_y *in MR tree* **then**

$$\mathcal{P}_\beta\left(\overbrace{m_x}^{w_i^{i+l}}\right) = \sum_h P(h|m_x) \times \mathcal{P}_\beta\left(\overbrace{m_y}^{w_j^k}\right)$$

        where (j, k) is determined by the hybrid sequence h.
      else if m_x *has two children* m_y *and* m_z *in MR tree* **then**

$$\mathcal{P}_\beta\left(\overbrace{m_x}^{w_i^{i+l}}\right) = \sum_h P(h|m_x) \times \mathcal{P}_\beta\left(\overbrace{m_y}^{w_m^n}\right) \times \mathcal{P}_\beta\left(\overbrace{m_z}^{w_p^q}\right)$$

        where (m, n) and (p, q) are both determined by the hybrid sequence h.
      else if m_x *has no child in MR tree* **then**

$$\mathcal{P}_\beta\left(\overbrace{m_x}^{w_i^{i+l}}\right) = P(m_x \rightarrow \overline{w_i\ w_{i+1}\ \ldots w_{i+l}})$$

      end
    end
  end
end

Algorithm 5.1: The conventional algorithm for computing the inside probabilities

Let us now consider the computation of inside probabilities with the efficient dynamic programming approach. Our goal is to compute the inside probability for the complete MR tree rooted by m_a that spans the complete sentence w_1^n. We have:

$$\mathcal{P}_\beta\left(\overbrace{m_a}^{w_1^n}\right) = \sum_r \phi\left(r|m_a\right) \times p\left(\left\langle \overbrace{m_a}^{w_1^n}, r \right\rangle\right) \tag{5.4}$$

where p term appearing in the right hand side of the above formula is the

aggregated inside probability for the N-M pair, where the hybrid pattern is restricted to r only.

The above formula can be extended to arbitrary word span and arbitrary MR production m_x, which refers to the aggregated inside probability for the MR production m_x covering the word sequence $w_s\ w_{s+1}\ \ldots\ w_l$:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x \\ \frown \\ w_s^l \end{array}\right) = \sum_r \phi\left(r|m_x\right) \times p\left(\left\langle \begin{array}{c} m_x \\ \frown \\ w_s^l \end{array}, r \right\rangle\right) \tag{5.5}$$

The p term in turn can be computed as follows. For example,

$$p\left(\left\langle \begin{array}{c} m_x \\ \frown \\ w_s^l \end{array}, m \rightarrow \mathcal{Y}\mathbf{w} \right\rangle\right) = \mathcal{P}_\beta\left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup\ \diagdown \\ \mathcal{M}_y \quad w_*^l \\ \frown \\ w_s^* \end{array}\right) \times \theta(\texttt{END}|m_x, w_l) \tag{5.6}$$

Now, the problem is reduced to computing the aggregated inside probabilities for aggregated hybrid trees.

For example, $\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup\ \diagdown \\ w_s^* \quad \mathcal{M}_y \\ \quad\ \frown \\ \quad\ w_*^l \end{array}\right)$ is defined as the aggregated inside probability score for all the hybrid trees rooted by m_a with context Λ_k that spans the NL word sequence $w_s\ w_{s+1}\ \ldots\ w_l$, and the hybrid pattern for all such hybrid trees below m_x is $m \rightarrow \mathbf{w}\mathcal{Y}$.

Given the definition of the aggregated hybrid tree, the inside probability associated with the above aggregated hybrid tree can be computed as follows:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \\ \quad\underline{\quad\quad} \\ \quad w_*^l \end{array}\right) = \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s \quad \boxed{\begin{array}{c} m_x(w_s) \\ | \\ \mathcal{M}_y \\ \underline{\quad\quad} \\ w_{s+1}^l \end{array}} \end{array}\right) + \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s \quad \boxed{\begin{array}{c} m_x(w_s) \\ w_{s+1}^* \quad \mathcal{M}_y \\ \quad\underline{\quad\quad} \\ \quad w_*^l \end{array}} \end{array}\right)$$

$$= \theta(w_s|m_x,\Lambda_k) \times \left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(w_s) \\ | \\ \mathcal{M}_y \\ \underline{\quad\quad} \\ w_{s+1}^l \end{array}\right) + \mathcal{P}_\beta\left(\begin{array}{c} m_x(w_s) \\ w_{s+1}^* \quad \mathcal{M}_y \\ \quad\underline{\quad\quad} \\ \quad w_*^l \end{array}\right)\right] \quad (5.7)$$

Note that in the above formula, the right hand side contains two other $\mathcal{P}_\beta$ terms. The term $\mathcal{P}_\beta\left(\begin{array}{c} m_x(w_s) \\ w_{s+1}^* \quad \mathcal{M}_y \\ \quad\underline{\quad\quad} \\ \quad w_*^l \end{array}\right)$ is in fact another problem whose size is smaller by 1 than the original problem (*i.e.* the term appearing in the left hand side of the formula). Thus, this smaller problem in turn can be computed by applying the above formula again.

As for the first $\mathcal{P}_\beta$ term, it can be computed with the following formula:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_u \\ \underline{\quad\quad} \\ w_s^l \end{array}\right) = \theta(\mathcal{M}_u|m_x,\Lambda_k) \times \rho(m_u|m_x,\arg_c) \times \mathcal{P}_\beta\left(\begin{array}{c} m_u \\ \underline{\quad\quad} \\ w_s^l \end{array}\right) \quad (5.8)$$

where m_u is the c-th child production of m_x under the child semantic category $\mathcal{M}_u$. Note that the $\mathcal{P}_\beta$ appearing in the right hand side is already defined above in Equation 5.5.

Similarly, as another example, for the aggregated hybrid tree that has the hybrid pattern $m \rightarrow \mathcal{Y}\mathbf{w}$ below the MR production m_x:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w_*^l \\ w_s^* \end{array}\right) = \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w_*^{l-1} \\ w_s^* \end{array}\right) \times \theta(w_l|m_x, w_{l-1})$$
$$+\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_y \\ w_s^{l-1} \end{array}\right) \times \theta(w_l|m_x, \mathcal{M}_y) \tag{5.9}$$

As for the aggregated hybrid tree that has the hybrid pattern $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}$ below the MR production m_x:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \quad w_*^l \\ w+ \end{array}\right) = \theta(w_s|m_x, \Lambda_k)$$
$$\times\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(w_s) \\ \mathcal{M}_y \quad w_*^l \\ w_{s+1}^* \end{array}\right) + \mathcal{P}_\beta\left(\begin{array}{c} m_x(w_s) \\ w_{s+1}^* \quad \mathcal{M}_y \quad w_*^l \\ w+ \end{array}\right)\right] \tag{5.10}$$

If there are two child semantic categories for a MR production, we can also establish similar relations. For the case where the hybrid pattern is $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}$:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \quad w_*^l \\ w+ \qquad\qquad w+ \end{array}\right)$$
$$= \sum_{s+2\le i\le l-2}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \\ w_*^{i-1} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ w_i^* \quad \mathcal{M}_z \quad w_*^l \\ w+ \end{array}\right)\right] \tag{5.11}$$

Here is another example for the case where the hybrid pattern is $m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z}$:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad | \quad \diagdown \\ \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \\ \overbrace{\quad}^{} \quad\quad \overbrace{\quad}^{} \\ w_s^* \quad\quad w_*^l \end{array}\right) = \sum_{s+1\leq i\leq l-1}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_y \\ \overbrace{\quad}^{} \\ w_s^{i-1} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \diagup \quad \diagdown \\ w_i^* \quad \mathcal{M}_z \\ \quad\quad \overbrace{\quad}^{} \\ \quad\quad w_*^l \end{array}\right)\right] \tag{5.12}$$

For a complete list of formulas explaining the relations amongst the terms for inside probability computation, please refer to Appendix B.

The Top-k Decoding Algorithm and the Viterbi Algorithm

So far, our focus on this chapter about the algorithms are for the training purpose. In other words, we are assuming that the N-M pairs are available and we perform parameter estimation via the EM algorithm. Besides the training algorithm, we also discuss other important algorithms related to the hybrid tree framework and the generative model. These important algorithms are analogous to the training algorithm for computing inside probabilities, and will be used for performing tasks such as semantic parsing and language generation, as to be discussed in later chapters.

We first consider the top-k decoding algorithm, which takes in an NL sentence and attempts to find the top k most probable hybrid trees that contain the NL sentence as its yield. The algorithm works analogously to the computation of inside probabilities with a CKY-style parsing chart. The algorithm scores a single probability score for each possible given N-M pair. However, this decoding algorithm maintains a stack of size k that keeps the top k most probable hybrid trees rooted by an MR production that covers the particular NL subsequence span under consideration. The algorithm performs in a bottom-up manner that guarantees the top k globally most probable hybrid trees that covers the complete NL sentence can be returned.

The other important algorithm is the Viterbi algorithm. The Viterbi algo-

rithm takes in a given N-M pair together with the learned model parameters, and returns the single most probable hybrid tree associated with the N-M pair. The algorithm is highly similar to the training algorithm for computing inside probabilities. The only difference is that instead of computing the sum over all possible different hybrid trees for a certain MR production - NL subsequence pair, we find the most single probable hybrid tree for that pair. This algorithm can help produce the pseudo-gold-standard hybrid trees for the training corpus, and is extremely important for both semantic parsing and language generation.

5.2.3 Computing Outside Probabilities

We use $\mathcal{P}_\alpha$ to denote the outside probability for a certain decomposed N-M pair. The standard procedure for computing the outside probabilities is given as Algorithm 2.

Analogous but more complex formulas are used for computing the outside probabilities. Updating of parameters can be incorporated into the computation of outside probabilities efficiently.

Similar to the notation used for computing the inside probabilities, we introduce the $\mathcal{P}_\alpha$ terms. For example, the following term

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ w_s^* \quad \mathcal{M}_y \\ \qquad \diagup \quad \diagdown \\ \qquad\quad w_*^l \end{array} \right) \tag{5.13}$$

refers to the probability score associated with all such hybrid trees in which the MR production m_x with context Λ_k spans the NL word sequence w_s^l, and expecting a hybrid pattern $m \rightarrow \mathbf{w}\mathcal{Y}$.

Data: Training set $\widetilde{\mathbf{w}}^d, \widetilde{\mathbf{m}}^d$ for $d = 1, \ldots, n$
Result: The outside probabilities for each N-M pair
$L = |\widetilde{\mathbf{m}}^d|$;
for *each MR production* m_x **do**

$$\mathcal{P}_\alpha\left(\overbrace{m_x}^{w_1^L}\right) = \begin{cases} 1 & \text{if } m_x \text{ is the root MR production in } \widetilde{\mathbf{m}}^d \\ 0 & \text{otherwise} \end{cases}$$

end
for $l \leftarrow L-1$ *to* 0 **do**
  for $i \leftarrow 1$ *to* $L-l$ **do**
    for m_x *as the next MR production in* $\widetilde{\boldsymbol{m}}^d$ *with pre-order traversal order of* $\widetilde{\boldsymbol{m}}^d$ **do**
      if m_x*'s parent* m_y *has no other child except* m_x*in MR tree* **then**

$$\mathcal{P}_\alpha\left(\overbrace{m_x}^{w_i^{i+l}}\right) = \sum_h P(h|m_y) \times \mathcal{P}_\alpha\left(\overbrace{m_y}^{w_j^k}\right)$$

        where (j,k) is determined by the hybrid sequence h.
      else if m_x*'s parent* m_y *has another child* m_z *in MR tree* **then**

$$\mathcal{P}_\alpha\left(\overbrace{m_x}^{w_i^{i+l}}\right) = \sum_h P(h|m_x) \times \mathcal{P}_\alpha\left(\overbrace{m_y}^{w_m^n}\right) \times \mathcal{P}_\beta\left(\overbrace{m_z}^{w_p^q}\right)$$

        where (m,n) and (p,q) are both determined by the hybrid sequence h.
      end
    end
  end
end

Algorithm 5.2: The conventional algorithm for computing the outside probabilities

An Improved Process

For the ease of discussion, we introduce an new operation $u \twoheadrightarrow v$ which is equivalent to

$$v \doteq \begin{cases} u & v \text{ is not initialized} \\ u+v & \text{otherwise} \end{cases} \tag{5.14}$$

Consider we have already computed the outside probability for the MR pro-

duction m_x with the NL span $w_s \dots w_l$. It is not hard to observe the following relation:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ w_s \quad \mathcal{M}_y \\ \quad \underset{w_{s+1}^l}{\frown} \end{array} \right) \times \theta(w_s | m_x, \Lambda_k) \times \theta(\mathcal{M}_y | m_x, w_s)$$
$$\times \rho(m_y | m_x, \arg_1) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_y(\texttt{BEGIN}) \\ \underset{w_{s+1}^l}{\frown} \end{array} \right) \tag{5.15}$$

where the MR production m_y with semantic category $\mathcal{M}_y$ is the first child for the MR production m_x.

In other words, we can collect the outside probabilities with an ***aggregation approach*** in a top-down manner. Note that this process is efficient, given the fact that it visits each N-M pair and each decomposed N-M pair exactly once.

Collecting Outside Probabilities

The algorithm for collecting outside probabilities follows a top-down process.

First of all, the outside probability for the complete MR tree rooted by m_a and the complete sentence $w_1 \dots w_n$ is 1:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_a \\ \underset{w_1^n}{\frown} \end{array} \right) = 1 \tag{5.16}$$

In general, we have:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x \\ \underset{w_s^l}{\frown} \end{array} \right) \times \phi(r | m_x) \twoheadrightarrow q \left(\left\langle \begin{array}{c} m_x \\ \underset{w_s^l}{\frown} \end{array}, r \right\rangle \right) \tag{5.17}$$

The q term can in turn be computed as follows, for example:

$$q\left(\left\langle \begin{array}{c} m_x \\ \wedge \\ w_s^l \end{array}, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w} \right\rangle\right) \times \theta(\texttt{END}|m_x, w_l) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \; | \; \diagdown \\ w_s^* \quad \mathcal{M}_y \quad w_*^l \\ \quad \wedge \\ \quad w+ \end{array} \right) \tag{5.18}$$

Now let us see how to handle the $\mathcal{P}_\alpha$ terms involving aggregated hybrid trees. For example,

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ w_s^* \quad \mathcal{M}_y \\ \quad \wedge \\ \quad w_*^l \end{array} \right) \times \theta(w_s|m_x, \Lambda_k) \twoheadrightarrow \left\{ \begin{array}{l} \mathcal{P}_\alpha \left(\begin{array}{c} m_x(w_s) \\ \diagup \quad \diagdown \\ w_{s+1}^* \quad \mathcal{M}_y \\ \quad \wedge \\ \quad w_*^l \end{array} \right) \\ \\ \mathcal{P}_\alpha \left(\begin{array}{c} m_x(w_s) \\ | \\ \mathcal{M}_y \\ \wedge \\ w_{s+1}^l \end{array} \right) \end{array} \right. \tag{5.19}$$

The above formula holds because if we are expecting a hybrid pattern $m \rightarrow \mathbf{w}\mathcal{Y}$, there are only two possibilities for the hybrid sequence below m_a. The first case is that it consists of a single NL word followed by the semantic category $\mathcal{M}_y$ (upper part of the equation), and the second is that it consists of more than 1 NL words followed by the semantic category (lower part of the equation).

Again, the upper $\mathcal{P}_\alpha$ term appearing in the right hand side of the above equation can be handled in a similar way as the original term appearing in the left hand side. The lower $\mathcal{P}_\alpha$ term can be computed as follows:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_u \\ \wedge \\ w_s^l \end{array} \right) \times \theta(\mathcal{M}_y|m_x, \Lambda_k) \times \rho(m_u|m_x, \arg_c) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_u \\ \wedge \\ w_s^l \end{array} \right) \tag{5.20}$$

where m_u is the c-th child of m_x.

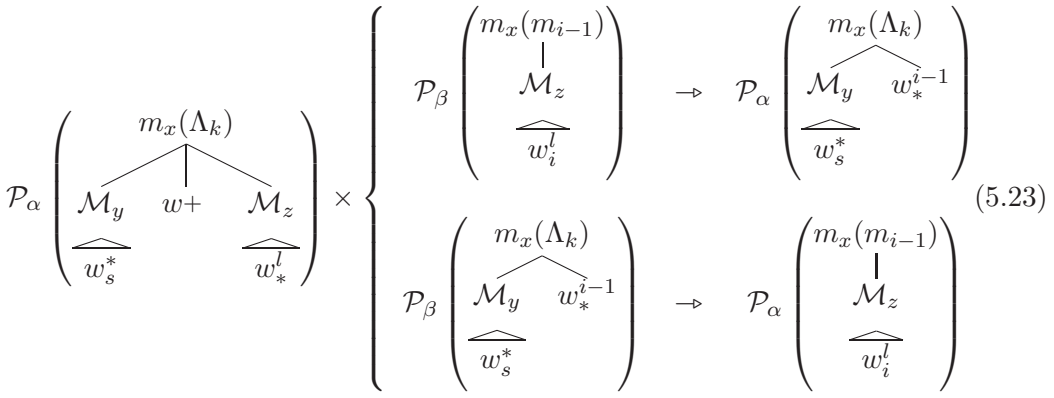
Here comes another example, where we are expecting the hybrid pattern $m \rightarrow \mathcal{YZ}$, for all $s+1 \leq i \leq l$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad \mathcal{M}_z \\ w_s^* \quad w_*^l \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \\ w_i^l \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \\ w_s^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \\ w_s^{i-1} \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \\ w_i^l \end{array}\right) \end{array} \right. \tag{5.21}$$

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{YZ}$, for all $s+2 \leq i \leq l$,we have:

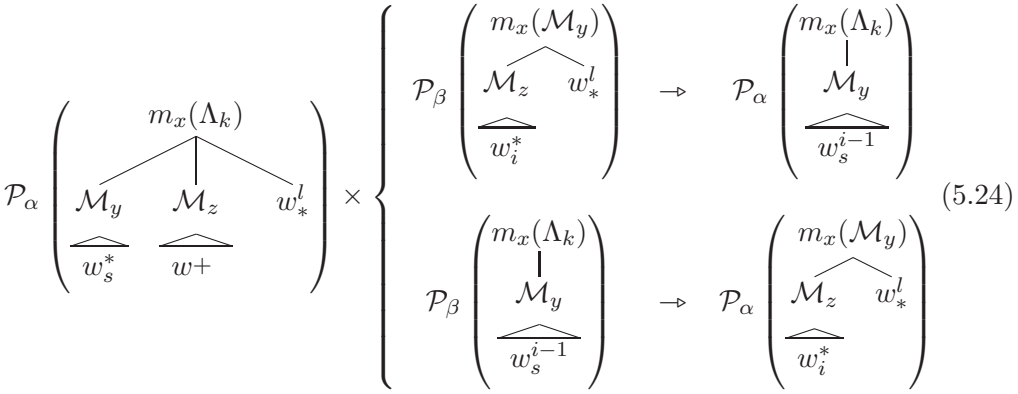
$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \quad \mathcal{M}_z \\ \quad w+ \quad w_*^l \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \\ w_i^l \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \\ \quad w_*^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \\ \quad w_*^{i-1} \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \\ w_i^l \end{array}\right) \end{array} \right. \tag{5.22}$$

For the case of expecting a pattern of the form $m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z}$, for all $s+2 \leq i \leq l$,we have:



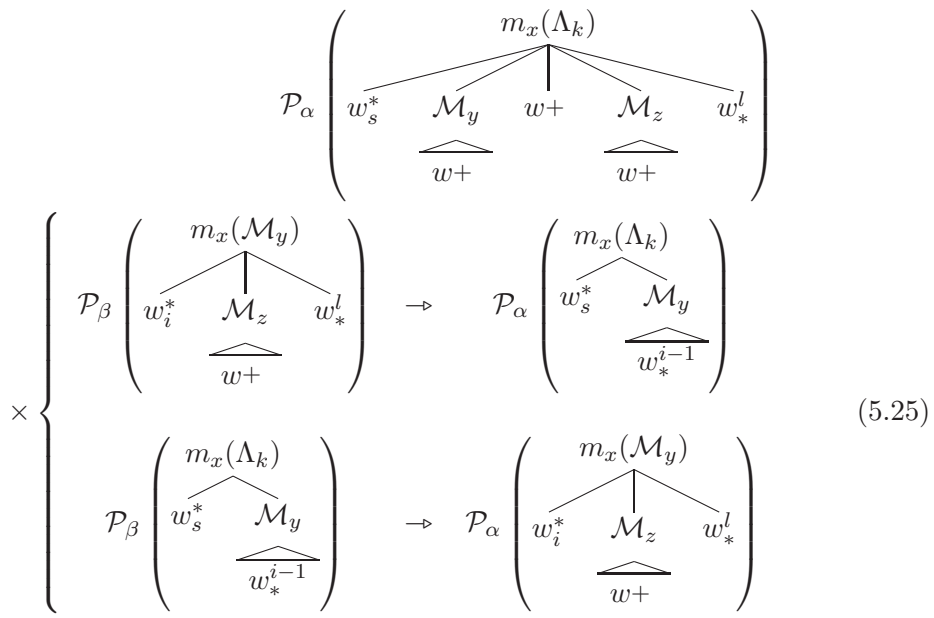
(5.23)

For the case of expecting a pattern of the form $m \rightarrow \mathcal{Y}\mathcal{Z}\mathbf{w}$, for all $s+1 \leq i \leq l-1$,we have:



(5.24)

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}$, for all $s+2 \leq i \leq l-2$,we have:



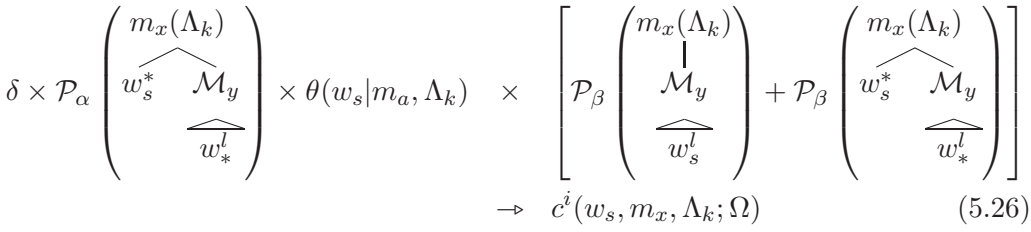
(5.25)

For a more complete list of formulas describing such relations, please refer to Appendix C.

Updating Model Parameters

Updating of the model's parameter can also be done using the idea of aggregation approach. In practice, this stage can be coupled with the process of computation for outside probabilities.

For example,



(5.26)

where

$$\delta = 1/\mathcal{P}_\beta\left(d^i\right) \tag{5.27}$$

is the normalization factor.

5.3 An Alternative View of the Algorithm

In this section, we argue that the discussed dynamic programming algorithm can be replaced with an equivalent PCFG rule set, in which each of the rule has one of the following form:

- $X \rightarrow Y$
- $X \rightarrow Y\,Z$
- $X \rightarrow t$

where X, Y and Z are all non-terminal symbols, and t is a terminal symbol.

The elegant property of the above PCFG rule set enables the algorithm to run in cubic time.

For the ease of discussion, we make unigram assumption throughout our discussion in this section. The PCFG rule set for the case of bigram assumption can be derived in an analogous way but is much more complicated.

Now let us see how to develop a PCFG rule set that generates the following NL word sequence from the MR production m_a:

$$m_a \rightarrow w_1\ w_2 \ldots\ w_{10}$$

The PCFG rule set consists of the following:

1. The PCFG rules that model the hybrid patterns.

$$m_a \rightarrow m_a^r\ \texttt{END} \tag{5.28}$$

for each possible hybrid pattern r. The symbols `END` is attached to the hybrid pattern.

Each such rule is assigned a probability score modeled by the pattern parameters. For example, the rule $m_a \rightarrow m_a^{\mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\ \texttt{END}}$ is assigned the probability score $\phi(m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}|m_a)$.

2. The PCFG rule that deals with the `END` symbol.

   There exists one PCFG rule for each possible hybrid pattern r. For example,

$$m_a^{\mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\ \texttt{END}} \rightarrow m_a^{\mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}} \tag{5.29}$$
$$m_a^{\mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}\ \texttt{END}} \rightarrow m_a^{\mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}} \tag{5.30}$$

   each of the above is assigned the probability score $\theta(\texttt{END}|m_a)$.

3. The PCFG rules that split the hybrid sequence.

   These rules model how to convert the left-hand side into two right-hand sides. Here are some example PCFG rules:

$$m_a^{\mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}} \rightarrow m_a^{\mathbf{w}\mathcal{Y}}\ m_a^{\mathbf{w}\mathcal{Z}} \tag{5.31}$$
$$m_a^{\mathbf{w}\mathcal{Y}} \rightarrow m_a^{w}\ m_a^{\mathbf{w}\mathcal{Y}} \tag{5.32}$$
$$m_a^{\mathbf{w}\mathcal{Y}} \rightarrow m_a^{w}\ m_a^{\mathcal{Y}} \tag{5.33}$$
$$m_a^{\mathbf{w}\mathcal{Z}} \rightarrow m_a^{w}\ m_a^{\mathbf{w}\mathcal{Z}} \tag{5.34}$$
$$m_a^{\mathbf{w}\mathcal{Z}} \rightarrow m_a^{w}\ m_a^{\mathcal{Z}} \tag{5.35}$$

   All such rules are assigned the probability 1.0.

4. The PCFG rules that model the emission probabilities.

$$m_a^{w} \rightarrow w_1 \tag{5.36}$$
$$m_a^{w} \rightarrow w_2 \tag{5.37}$$
$$\vdots$$
$$m_a^{w} \rightarrow w_{10} \tag{5.38}$$

Each such rule is assigned the model's emission probability. For example, the rule $m_a^w \rightarrow w_1$ is assigned the probability $\theta(w_1|m_a, \Lambda)$, where Λ is empty in this case.

It is important to note that the above probabilistic PCFG rule set is deficient, given the fact that the probability scores for the rules with common left-hand side nonterminal do not always sum up to 1. However, this PCFG rule set is only an auxiliary rule set used to reduce the computation power for the parameter estimation phase and are not the actual PCFG rules used by the generative model.

5.4 Algorithmic Complexity Analysis

In this section, we give algorithmic analysis for the dynamic programming algorithm presented in this chapter, and show why the algorithm successfully reduce the time complexity from $O(n^6m)$ to $O(n^3m)$.

First we show why the naive algorithm requires $O(n^6m)$ time. Consider parsing a sentence using the CKY-style parse chart, with the following hybrid pattern:

$$m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathrm{Z}\mathbf{w}$$

Covering any NL subsequence span with this hybrid pattern will require splitting the subsequence into 5 segments. Each such segmentation requires $O(1)$ running time. On the other hand, for a given NL sentence, we need to perform the segmentation for all possible such subsequence spans. In other words, for any NL sentence of length n, the running time is proportional to the number of all possible ways to split it into 7 contiguous segments. Thus, the running time is equivalent to the number of possible ways to insert 6 split points inside the sentence of length n, which is:

$$\binom{n}{6} \in O(n^6) \tag{5.39}$$

Note that according to the list of hybrid patterns listed in Table 4.1, the above example hybrid pattern produces the maximum number of segmentation for a given subsequence. Computation of the outside probabilities is similar to the computation of inside probabilities and should require the same time complexity. Therefore it gives the upper bound to the time complexity. For an N-M pair with m MR productions, the total time complexity of the naive algorithm will therefore require $O(n^6 m)$ time for one EM iteration.

We focus our discussion on the computation of inside probabilities first. The computation of inside probabilities are performed with a bottom-up approach. First of all, the computation of the following formula will be performed once only for each possible m_x and (s, l):

$$\mathcal{P}_\beta \left(\begin{array}{c} m_x \\ \frown \\ w_s^l \end{array} \right) = \sum_r \phi\left(r|m_x\right) \times p \left(\left\langle \begin{array}{c} m_x \\ \frown \\ w_s^l \end{array}, r \right\rangle \right) \tag{5.40}$$

The above step only requires $O(1)$ time to compute. On the other hand, the term $p\left(\left\langle \begin{array}{c} m_x \\ \frown \\ w_s^l \end{array}, r \right\rangle \right)$ can be rewritten as expressions that involve other $\mathcal{P}_\beta$ terms. For example if we assume the MR production m_x has one single child semantic category, we have:

$$p \left(\left\langle \begin{array}{c} m_x \\ \frown \\ w_s^l \end{array}, m \rightarrow \mathcal{Y}\mathbf{w} \right\rangle \right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagdown \\ \mathcal{M}_y \qquad w_*^l \\ \frown \\ w_s^* \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{5.41}$$

Let us denote the total time taken before computing a certain formula f as

$t_\beta(f)$. Consider a specific MR production m_a, we have:

$$t_\beta\left(\left\langle \begin{array}{c} m_a \\ \wedge \\ w_s^l \end{array}, m \to \mathcal{Y}\mathbf{w} \right\rangle\right) = t_\beta\left(\begin{array}{c} m_a(\texttt{BEGIN}) \\ \mathcal{M}_b \quad w_*^l \\ \wedge \\ w_s^* \end{array}\right) + O(1) \tag{5.42}$$

On the other hand, we have:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_a(\Lambda_k) \\ \mathcal{M}_b \quad w_*^l \\ \wedge \\ w_s^* \end{array}\right) = \mathcal{P}_\beta\left(\begin{array}{c} m_a(w_s) \\ \mathcal{M}_b \quad w_*^{l-1} \\ \wedge \\ w_s^* \end{array}\right) \times \theta(w_l|m_a, w_{l-1})$$
$$+\mathcal{P}_\beta\left(\begin{array}{c} m_a(w_s) \\ | \\ \mathcal{M}_b \\ \wedge \\ w_1^{l-1} \end{array}\right) \times \theta(w_l|m_a, \mathcal{M}_b) \tag{5.43}$$

Since the dynamic programming algorithm performs in a bottom-up manner, we note that the two $\mathcal{P}_\beta$ terms appearing in the right-hand side of the above equation has already been computed. Therefore the above equation requires $O(1)$ time complexity.

Therefore, consider a particular MR production m_a and a certain NL sentence with n words, the total time complexity for this series of formulas is:

$$t_\beta[m_a, m \to \mathcal{Y}\mathbf{w}] = \sum_{l=2}^{n}\sum_{s=1}^{l-1} 1 = \sum_{l=2}^{n}(l-1) = \frac{n(n-1)}{2} \in O(n^2) \tag{5.44}$$

Similarly, for all the possible hybrid patterns listed in Table 4.1 with 1 right-hand side nonterminal symbol, we have the time complexity $O(n^2)$. Thus, the total time complexity for computing the inside probability for all possible N-M pairs that involve the MR production m_a is $O(n^2)$.

Now let us consider another MR production who has two child semantic categories. We have:

$$p\left(\left\langle \overset{m_a}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w} \right\rangle\right)$$

$$= \mathcal{P}_\beta \left(\begin{array}{ccccccc} & & & m_a(\texttt{BEGIN}) & & & \\ & & & \diagup \quad | \quad \diagdown & & & \\ w_s^* & & \mathcal{M}_b & w+ & \mathcal{M}_c & & w_*^l \\ & & \underset{w+}{\triangle} & & \underset{w+}{\triangle} & & \end{array} \right) \times \theta(\texttt{END}|m_a, w_l) \tag{5.45}$$

The $\mathcal{P}_\beta$ term appearing in the right hand side can be rewritten as:

$$\mathcal{P}_\beta \left(\begin{array}{ccccccc} & & & m_a(\Lambda) & & & \\ & & & \diagup \quad | \quad \diagdown & & & \\ w_s^* & & \mathcal{M}_b & w+ & \mathcal{M}_c & & w_*^l \\ & & \underset{w+}{\triangle} & & \underset{w+}{\triangle} & & \end{array} \right)$$

$$= \sum_{s+2 \leq i \leq l-2} \left[\mathcal{P}_\beta \left(\begin{array}{ccccc} & & m_a(\Lambda) & & \\ & & \diagup \quad \diagdown & & \\ w_s^* & & \mathcal{M}_b & & w_*^i \\ & & \underset{w+}{\triangle} & & \end{array} \right) \times \mathcal{P}_\beta \left(\begin{array}{ccc} & m_a(w_i) & \\ & \diagup \quad \diagdown & \\ \mathcal{M}_c & & w_*^l \\ \underset{w_{i+1}^*}{\triangle} & & \end{array} \right) \right] \tag{5.46}$$

Note that due to the behavior of the dynamic programming, the terms appearing in the right hand side of the equation are all computed already. Thus the time for computing this equation has the complexity $O(l-s)$.

Mathematically, we have:

$$t_\beta[m_a, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}, s, l] \in O(l-s) \tag{5.47}$$

Thus, the total time complexity for computing the inside probabilities under the MR production m_a is:

$$t_\beta[m_a, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}] \in O\left(\sum_{l=2}^{n}\sum_{s=1}^{l-1}(l-s)\right) \equiv O(n^3) \tag{5.48}$$

Therefore, the time complexity for computing the inside probabilities for all

possible N-M pairs with subtree rooted by m_a is $O(n^3)$. Consequently, in general, the time complexity for computing all the inside probabilities for the complete MR tree with m MR productions should be $O(n^3 m)$.

The complexity for computation of outside probabilities can be analyzed in an analogous way but is less intuitive.

First we consider the following initial step for collecting outside probabilities:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_u \\ \frown \\ w_s^l \end{array} \right) \times \theta(\mathcal{M}_y | m_x, \Lambda_k) \times \rho(m_u | m_x, \arg_c) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_u \\ \frown \\ w_s^l \end{array} \right) \tag{5.49}$$

where m_u is the c-th child of m_x.

For any given MR production m_a, the time complexity associated with this computation step is $O(1)$ only. The newly collected outside probability term to the right of the above formula can be further processed as discussed in Section 5.2.3. Let us consider the most complicated case - a case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}$, for all $s+2 \leq i \leq l-2$,we have:

$$\mathcal{P}_\alpha \left(\begin{array}{ccccc} & & m_a(\Lambda_k) & & \\ & \swarrow & \swarrow & \downarrow & \searrow \\ w_s^* & & \mathcal{M}_b & w+ & \mathcal{M}_c & & w_*^l \\ & & \frown & & \frown & & \\ & & w+ & & w+ & & \end{array} \right)$$

$$\times \left\{ \begin{array}{l} \mathcal{P}_\beta \left(\begin{array}{ccccc} & & m_a(\mathcal{M}_b) & & \\ & \swarrow & \downarrow & \searrow & \\ w_i^* & & \mathcal{M}_c & & w_*^l \\ & & \frown & & \\ & & w+ & & \end{array} \right) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{ccc} & m_a(\Lambda_k) & \\ & \swarrow \quad \searrow & \\ w_s^* & & \mathcal{M}_b \\ & & \frown \\ & & w_*^{i-1} \end{array} \right) \\ \\ \mathcal{P}_\beta \left(\begin{array}{ccc} & m_a(\Lambda_k) & \\ & \swarrow \quad \searrow & \\ w_s^* & & \mathcal{M}_b \\ & & \frown \\ & & w_*^{i-1} \end{array} \right) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{ccccc} & & m_a(\mathcal{M}_b) & & \\ & \swarrow & \downarrow & \searrow & \\ w_i^* & & \mathcal{M}_c & & w_*^l \\ & & \frown & & \\ & & w+ & & \end{array} \right) \end{array} \right. \tag{5.50}$$

Each such step requires $O(1)$ operation, and the total time complexity for performing all the above operations is:

$$t_\alpha[m_a, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}, l, s] \in O(l-s) \tag{5.51}$$

For an NL sentence with n NL words, the total time complexity is:

$$t_\alpha[m_a, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}, l, s] \in O\left(\sum_{l=2}^{n}\sum_{s=1}^{l-1}(l-s)\right) \equiv O(n^3) \tag{5.52}$$

This means that computation of the outside probability for a particular MR production m_a where the above hybrid pattern is expected requires $O(n^3)$ time. Note that analogous to the case of computing inside probabilities, the two new terms introduced by Equation 5.50 are in fact each expecting a simpler hybrid pattern than the original term for their subsequent processing, which requires $O(n^2)$ time to perform subsequent processing. Thus, the total time complexity for the computation of all outside probabilities is again $O(n^3m)$.

In conclusion, the time complexity for the proposed dynamic programming algorithm requires $O(n^3m)$ time for handling one particular N-M pair in one EM iteration, where n is the number of NL words in the NL sentence, and m is the number of MR productions in the MR tree.

Chapter 6

Semantic Parsing and Language Generation with Hybrid Trees

In this chapter, we discuss the applications of the generative models built on top of our proposed hybrid tree framework. Specifically, we consider two problems, namely ***semantic parsing***, as well as ***language generation***. The former task is to transform a natural language sentence to its corresponding meaning representation, while the latter is to generate a natural language sentence that explains the meaning conveyed by a particular meaning representation.

6.1 Semantic Parsing with Hybrid Tree

Semantic parsing is the task of transforming natural language to corresponding meaning representations. We first discuss how to make use of the generative model to directly solve the this task, and followed by some more sophisticated methods to enhance the performance.

6.1.1 Base Model

After we have learned the parameters for the generative model, we can make use of the learned parameters to perform the task of transforming natural language to meaning representations.

In this task, we want to find the optimal MR structure $\widetilde{\mathbf{m}}^*$ given a new NL sentence $\widetilde{\mathbf{w}}$:

$$\widetilde{\mathbf{m}}^* = \arg\max_{\widetilde{\mathbf{m}}} P(\widetilde{\mathbf{m}}|\widetilde{\mathbf{w}}) \tag{6.1}$$

Since each N-M pair is associated with many possible hybrid trees, we have:

$$\widetilde{\mathbf{m}}^* = \arg\max_{\widetilde{\mathbf{m}}} \sum_{\mathcal{T}} P(\widetilde{\mathbf{m}}, \mathcal{T}|\widetilde{\mathbf{w}}) \tag{6.2}$$

where $\mathcal{T}$ is a possible hybrid tree associated with the $\widetilde{\mathbf{m}}$-$\widetilde{\mathbf{w}}$ pair. However, it is expensive to compute the summation over all possible hybrid trees. We therefore find the most likely hybrid tree instead:

$$\widetilde{\mathbf{m}}^* \approx \arg\max_{\widetilde{\mathbf{m}}} \max_{\mathcal{T}} P(\widetilde{\mathbf{m}}, \mathcal{T}|\widetilde{\mathbf{w}}) = \arg\max_{\widetilde{\mathbf{m}}} \max_{\mathcal{T}} P(\widetilde{\mathbf{w}}, \widetilde{\mathbf{m}}, \mathcal{T}) \tag{6.3}$$

This formula says that the most probable MR tree is obtained from the hybrid tree $\mathcal{T}$ that gives the maximal probability score $P(\widetilde{\mathbf{w}}, \widetilde{\mathbf{m}}, \mathcal{T})$, over all possible hybrid trees that contain the given NL sentence $\widetilde{\mathbf{w}}$.

Thus, the process is as follows: given a new NL sentence $\widetilde{\mathbf{w}}$, we find the most probable hybrid tree $\mathcal{T}$ that contains the NL sentence $\widetilde{\mathbf{w}}$, such that the probability for generating the hybrid tree is maximized. The MR tree contained by the most probable hybrid tree $\mathcal{T}$ is exactly the MR tree that we are looking for.

The algorithm for finding the most probable hybrid tree given a new NL sentence can also be done with a bottom-up approach with a CKY parsing chart. Similar dynamic programming algorithm as the one presented in Chapter 5 for mode training can also be applied in this phase.

In practice, the single top ranked hybrid tree may not always contain the correct MR tree. Rather, we found that the correct MR tree always appears amongst the top ranked hybrid trees, but may not always appear in highest scoring hybrid tree.

We have implemented an exact top-k decoding algorithm for this task. Dynamic programming techniques similar to those discussed in the last chapter can also be applied when retrieving the top candidates.

We also find the Viterbi hybrid tree given an N-M pair, which can be done in an analogous way. This tree will be useful for discriminative reranking, as to be discussed next.

6.1.2 Reranking and Filtering of Predictions

Due to the various independence assumptions we have made, the model lacks the ability to express some long range dependencies. We therefore post-process the best candidate predictions with a discriminative reranking algorithm.

The Averaged Perceptron Algorithm with Separating Plane

The averaged perceptron algorithm [Col02] has previously been applied to various NLP tasks [Col02, Col01] for discriminative reranking. The detailed algorithm can be found in [Col02]. In this section, we extend the conventional averaged perceptron by introducing an explicit separating plane on the feature space.

Our reranking approach requires three components during training: a **GEN** function that defines for each NL sentence a set of candidate hybrid trees; a single correct reference hybrid tree for each training instance; and a feature function Φ that defines a mapping from a hybrid tree to a feature vector. The algorithm learns a weight vector $\mathbf{w}$ that associates a weight to each feature, such that a score $\mathbf{w} \cdot \Phi(\mathcal{T})$ can be assigned to each candidate hybrid tree $\mathcal{T}$. Given a new instance, the hybrid tree with the highest score is then picked by the algorithm

as the output.

In this task, the **GEN** function is defined as the output hybrid trees of the top-k (k is set to 50 in our experiments) decoding algorithm, given the learned model parameters. The "correct" reference hybrid tree is approximated by running the Viterbi algorithm on each training N-M pair. The feature function is discussed next.

While conventional perceptron algorithms usually optimize the accuracy measure, we extend it to allow optimization of the F-measure by introducing an explicit separating plane on the feature space that rejects certain predictions even when they score highest. The idea is to find a threshold b after $\mathbf{w}$ is learned, such that a prediction with score below b gets rejected. We pick the threshold that leads to the optimal F-measure when applied to the training set. The complete training and testing algorithms are given in Algorithm 3 and 4 respectively.

Figure 6.1 gives an graphical view of the conventional perceptron and this extension. As illustrated, after the conventional perceptron is trained, for each instance we leave on the space only the highest scoring candidate data points and remove the rest. All the data points to the right of the separating plane will be considered as predicted, while the points to the left will be considered as non-predicted. The F-measure of the training set is evaluated based on such a separation.

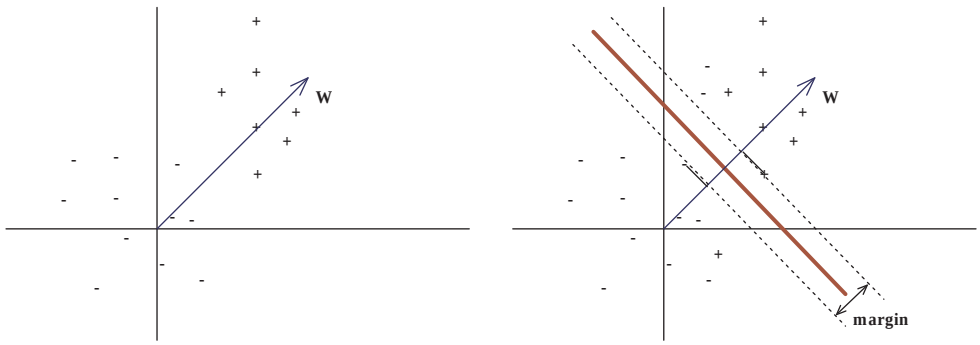


Figure 6.1: The conventional perceptron vs. the perceptron with separating plane

Data: Training set x_i, y_i for $i = 1, \ldots, n$
Result: The weight vector $\mathbf{w}$, as well as the prediction threshold b
Initialize $m = 0$, $\mathbf{w}^0 = \mathbf{0}$;
for $t = 1, \ldots, T, i = 1, \ldots, n$ **do**
  $z_i = \text{argmax}_{y \in \mathbf{GEN}(x_i)} \mathbf{f}(x_i, y) \cdot \mathbf{w}^m$;
  if $z_i \neq y_i$ **then**
    $m = m + 1$;
    $\mathbf{w}^m = \mathbf{w}^{m-1} + \mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, z_i)$;
  end
end
Compute the averaged weight vector: $\mathbf{w} = \left(\sum_{k=1}^{m} \mathbf{w}^k\right)/m$;
Normalize the weight vector: $\mathbf{w} := \mathbf{w}/|\mathbf{w}|$;
$r = n$;
for $i = 1, \ldots, n$ **do**
  $z_i = \text{argmax}_{y \in \mathbf{GEN}(x_i)} \mathbf{f}(x_i, y) \cdot \mathbf{w}$;
  $c_i = 1$;
  if $z_i \neq y_i$ **then**
    $c_i = 0$;
    $r := r - 1$;
    $v_i = \max_{y \in \mathbf{GEN}(x_i)} \mathbf{f}(x_i, y) \cdot \mathbf{w}$;
  end
  Store the pair $p_i \equiv \langle v_i, c_i \rangle$;
end
Sort the pairs p_i with ascending order based on the values of v_i;
$F_{max} = r/n$;
$b = v_1$;
for $i = 1, \ldots, n - 1$ **do**
  if $c_i \equiv 0$ **then**
    $r := r - 1$;
    if $c_{i+1} \equiv 1$ **then**
      $F = 2r/(2n - i)$;
      if $F > F_{max}$ **then**
        $F_{max} = F$;
        $b = (v_i + v_{i+1})/2$;
      end
    end
  end
end
return $(\mathbf{w}, b)$;

Algorithm 6.1: The averaged perceptron with separating plane (Training)

Input: Testing instance x, learned weight $\mathbf{w}$, learned threshold b
if $\max_{y \in \boldsymbol{GEN}(x)} \boldsymbol{f}(x, y) \cdot \boldsymbol{w} - b < 0$ **then**
  Do not predict
else
  Predict

$$P(x) = \underset{y \in \mathbf{GEN}(x)}{\text{argmax}} \ \mathbf{f}(x, y) \cdot \mathbf{w} - b$$

end

Algorithm 6.2: The averaged perceptron with separating plane (Testing)

Features

We describe in this section the set of features we used. Examples are given based on the hybrid tree in Figure 4.2. Some of the them are adapted from [CK05] for a natural language parsing task. Features 1-5 are indicator functions (*i.e.,* it takes value 1 if a certain combination as the ones listed below is present, 0 otherwise), while feature 6 is real valued. Features that do not appear more than once in the training set are discarded.

Hybrid rule features : one MR production and its child hybrid form.

For example,

$$f_1 : \textsc{State} : loc\_1(\textsc{River}) \rightarrow \texttt{have}\ \textsc{River}$$

Expanded hybrid rule features : one MR production and its child hybrid form expanded.

For example,

$$f_2 : \textsc{State} : loc\_1(\textsc{River}) \rightarrow \langle \texttt{have}, \textsc{River} : river(\text{all}) \rangle$$

Long-range unigram features : one MR production and an NL word appearing below in tree.

For example,

$$f_3 : \textsc{State} : exclude(\textsc{State}\ \textsc{State}) \rightarrow \texttt{rivers}$$

Grandchild unigram features : one MR production and its grandchild NL word.

For example,

$$f_4 : \textsc{State} : loc\_1(\textsc{River}) \rightarrow \texttt{rivers}$$

Two Level unigram features : one MR production, its parent production, and its child NL word.

For example,

$$f_5 : \langle \textsc{River} : river(\text{all}), \textsc{State} : loc\_1(\textsc{River}) \rangle \rightarrow \texttt{rivers}$$

Model log-probability feature : logarithm of base model's joint probability.

In other words,

$$\log \left(\widehat{P}(\mathbf{w}, \mathbf{m}, \mathcal{T}) \right)$$

6.2 Language Generation with Hybrid Tree

The task of generating NL sentences from MRs can be defined as follows. Given a training corpus consisting of MRs paired with their NL sentences, one needs to develop algorithms that learn how to effectively "paraphrase" MRs with natural language sentences. During testing, the system should be able to output the most probable NL "paraphrase" for a given new MR.

The generative system models $P(\mathcal{T}(\widetilde{\mathbf{w}}, \widetilde{\mathbf{m}}))$, the joint generative process for the hybrid tree containing both NL and MR. This term can be rewritten in the following way:

$$P(\mathcal{T}(\widetilde{\mathbf{w}}, \widetilde{\mathbf{m}})) = P(\widetilde{\mathbf{m}}) \times P\left(\mathcal{T}(\widetilde{\mathbf{w}}, \widetilde{\mathbf{m}}) | \widetilde{\mathbf{m}}\right) \tag{6.4}$$

In other words, we reach an alternative view of the joint generative process as follows. We choose to generate the complete MR $\widetilde{\mathbf{m}}$ first. Given $\widetilde{\mathbf{m}}$, we generate hybrid sequences below each of its MR production, which gives us a complete hybrid tree $\mathcal{T}(\widetilde{\mathbf{w}}, \widetilde{\mathbf{m}})$. The NL sentence $\widetilde{\mathbf{w}}$ can be constructed from this hybrid tree exactly.

We define an operation $yield(\mathcal{T})$ which returns the NL sentence as the yield of the hybrid tree $\mathcal{T}$. Given an MR $\widetilde{\mathbf{m}}$, we find the most probable NL sentence $\widetilde{\mathbf{w}}^*$ as follows:

$$\widetilde{\mathbf{w}}^* = yield\Big(\operatorname*{argmax}_{\mathcal{T}} P(\mathcal{T}|\widetilde{\mathbf{m}})\Big) \tag{6.5}$$

In other words, we first find the most probable hybrid tree $\mathcal{T}$ that contains the provided MR $\widetilde{\mathbf{m}}$. Next we return the yield of $\mathcal{T}$ as the most probable NL sentence.

Different assumptions can be made in the process of finding the most probable hybrid tree. We first describe a simple model which is a direct inversion of the semantic parser. This model, as a baseline model, generates a complete NL sentence word by word. Next, a more sophisticated model that exploits NL phrase-level dependencies is built that tackles some weaknesses of the simple baseline model.

6.2.1 Direct Inversion Model

Assume that a pre-order traversal of the MR $\widetilde{\mathbf{m}}$ gives us the list of MR productions $m_1, m_2, \ldots, m_S$, where S is the number of MR productions in $\widetilde{\mathbf{m}}$. Based on the independence assumption made by the generative models, each MR production independently generates a hybrid sequence. Denote the hybrid sequence

generated under the MR production m_s as h_s, for $s = 1, \ldots, S$. We call the list of hybrid sequences $\mathbf{h} = \langle h_1, h_2, \ldots, h_S \rangle$ a *hybrid sequence list* associated with this particular MR. Thus, our goal is to find the optimal hybrid sequence list $\mathbf{h}^*$ for the given MR $\widetilde{\mathbf{m}}$, which is formulated as follows:

$$\mathbf{h}^* = \langle h_1^*, \ldots, h_S^* \rangle = \underset{h_1,\ldots,h_S}{\operatorname{argmax}} \prod_{s=1}^{S} P(h_s|m_s) \tag{6.6}$$

The optimal hybrid sequence list defines the optimal hybrid tree whose yield gives the optimal NL sentence.

Due to the strong independence assumption introduced by the generative models, the hybrid tree generation process is in fact highly decomposable. Optimization of the hybrid sequence list $\langle h_1, \ldots, h_S \rangle$ can be performed individually since they are independent of one another. Thus, mathematically, for $s = 1, \ldots, S$, we have:

$$h_s^* = \underset{h_s}{\operatorname{argmax}}\, P(h_s|m_s) \tag{6.7}$$

We have applied three generative models for the task of semantic parsing. In this inverse task, for generation of a hybrid sequence, we choose to use the bigram model. We choose this model mainly due to its stronger ability in modeling dependencies between adjacent NL words, which we believe to be quite important in this NL generation task. With the bigram model assumption, the optimal hybrid sequence that can be generated from each MR production is defined as follows:

$$\begin{aligned} h_s^* &= \underset{h_s}{\operatorname{argmax}}\, P(h_s|m_s) \\ &= \underset{h_s}{\operatorname{argmax}} \left\{ \phi(r|m_s) \times \prod_{j=1}^{|h_s|+1} \theta(t_j|m_s, t_{j-1}) \right\} \end{aligned} \tag{6.8}$$

where t_i is either an NL word or a semantic category with $t_0 \equiv$ `BEGIN` and $t_{|h_s|+1} \equiv$ `END`, and r is the hybrid pattern that matches the hybrid sequence h_s,

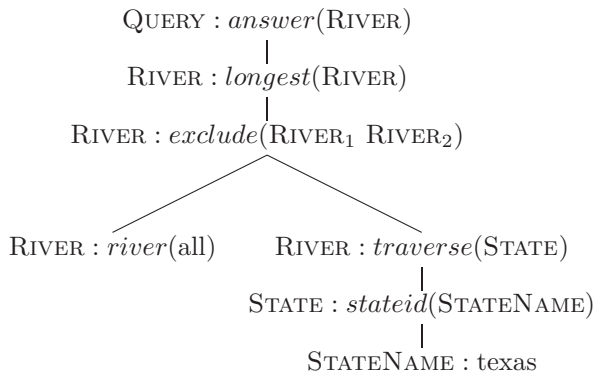
which is equivalent to $t_1, \ldots, t_{|h_s|}$.

Equivalently, we can view the problem in the log-space:

$$h_s^* = \operatorname*{argmin}_{h_s} \Bigg\{ -\log \phi(r|m_s) + \sum_{j=1}^{|h_s|+1} -\log \theta(t_j|m_s, t_{j-1}) \Bigg\} \tag{6.9}$$

Note the term $-\log \phi(r|m_s)$ is a constant for a particular MR production m_s and a particular hybrid pattern r. This search problem can be equivalently cast as the shortest path problem which can be solved efficiently with Dijkstra's algorithm [CLRS01]. We define a set of states. Each state represents a single NL word or a semantic category, including the special symbols `BEGIN` and `END`. A directed path between two different states t_u and t_v is associated with a distance measure $-\log \theta(t_v|m_s, t_u)$, which is non-negative. The task now is to find the shortest path between `BEGIN` and `END`. (In practice, we should make sure that the generated hybrid sequence $t_0 \ldots t_{|h_s|+1}$ is a valid hybrid sequence that comply with the hybrid pattern r. For example, the MR production STATE : *loc_1*(RIVER) can generate the following hybrid sequence "`BEGIN have` RIVER `END`" but not this hybrid sequence "`BEGIN have END`". This can be achieved by finding the shortest path from `BEGIN` to RIVER, which then gets concatenated to the shortest path from RIVER to `END`.) The sequence of words appearing in this path is simply the most probable hybrid sequence under this MR production m_s. We build this model by directly inverting the semantic parser, and this model is therefore referred to as the *direct inversion model*.

A major weakness of this baseline model is that it encodes strong independence assumptions during the hybrid tree generation process. Though shown to be effective in the task of transforming NL to MR, such independence assumptions may introduce difficulties in this NLG task. For example, consider the MR




```
what is the longest river that does not run through texas
```

Figure 6.2: An example MR paired with its NL sentence.

shown in Figure 6.2. The generation steps of the hybrid sequences from the two adjacent MR productions QUERY : *answer*(RIVER) and RIVER : *longest*(RIVER) are completely independent of each other. This may harm the fluency of the generated NL sentence, especially when a transition from one hybrid sequence to another is required. In fact, due to such an independence assumption, the model always generates the same hybrid sequence from the same MR production, regardless of its context such as parent or child MR productions. Such a limitation points to the importance of better utilizing the tree structure of the MR for this NLG task. Furthermore, due to the bigram assumption, the model is unable to capture longer range dependencies amongst the words or semantic categories in each hybrid sequence.

To tackle the above issues, we explore ways of relaxing various assumptions, which leads to an alternative model as discussed next.

6.2.2 A Tree Conditional Random Fields Model

Based on the belief that using known phrases usually leads to better fluency in the NLG task [WM07a], we explore methods for generating an NL sentence at phrase level rather than at word level. This is done by generating hybrid sequences as complete objects, rather than sequentially one word or semantic category at a

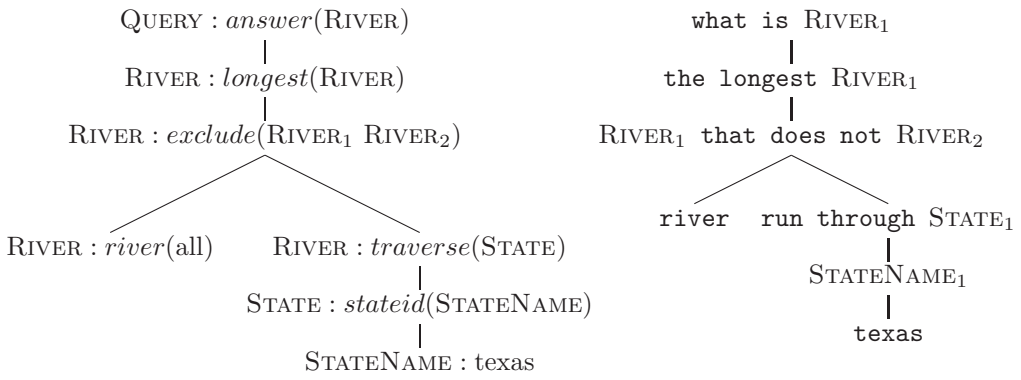


Figure 6.3: An MR (left) and its associated hybrid sequences (right)

time, from MR productions.

We assume that each MR production can generate a complete hybrid sequence below it from a finite set of possible hybrid sequences. Each such hybrid sequence is called a *candidate hybrid sequence* associated with that particular MR production. Given a set of candidate hybrid sequences associated with each MR production, the generation task is to find the optimal hybrid sequence list $\mathbf{h}^*$ for a given MR $\widehat{\mathbf{m}}$:

$$\mathbf{h}^* = \operatorname*{argmax}_{\mathbf{h}} p(\mathbf{h}|\widehat{\mathbf{m}}) \tag{6.10}$$

Figure 6.3 shows a complete MR, as well as a possible tree that contains hybrid sequences associated with the MR productions. For example, in the figure the MR production RIVER : *traverse*(STATE) is associated with the hybrid sequence `run through` STATE$_1$. Each MR production can be associated with potentially many different hybrid sequences. The task is to determine the most probable list of hybrid sequences as the ones appearing on the right of Figure 6.3, one for each MR production.

To make better use of the tree structure of MR, we take the approach of modeling the conditional distribution using a log-linear model. Following the conditional random fields (CRF) framework [LMP01], we can define the probability of the hybrid sequence list given the complete MR $\widehat{\mathbf{m}}$, as follows:

$$p(\mathbf{h}|\widehat{\mathbf{m}}) = \frac{1}{Z(\widehat{\mathbf{m}})} \exp\left(\sum_{i \in V} \sum_{k} \mu_k g_k(h_i, \widehat{\mathbf{m}}, i) + \sum_{(i,j) \in E} \sum_{k} \lambda_k f_k(h_i, h_j, \widehat{\mathbf{m}}, i, j) \right) \tag{6.11}$$

where V is the set of all the vertices in the tree, and E is the set of the edges in the tree, consisting of parent-child pairs. The function $Z(\widehat{\mathbf{m}})$ is the normalization function. Note that the dependencies among the features here form a tree, unlike the sequence models used in Lafferty et al. [LMP01]. The function $f_k(h_i, h_j, \widehat{\mathbf{m}}, i, j)$ is a feature function of the entire MR tree $\widehat{\mathbf{m}}$ and the hybrid sequences at vertex i and j. These features are usually referred to as the edge features in the CRF framework. The function $g_k(h_i, \widehat{\mathbf{m}}, i)$ is a feature function of the hybrid sequence at vertex i and the entire MR tree. These features are usually referred to as the vertex features. The parameters λ_k and μ_k are learned from the training data.

In this task, we are given only N-M pairs and do not have the hybrid tree corresponding to each N-M pair as training data. Now we describe how the set of candidate hybrid sequences for each MR production is obtained as well as how the training data for this model is constructed. After the joint generative model is learned [LNLZ08], we first use a Viterbi algorithm to find the optimal hybrid tree for each MR-NL pair in the training set. From each optimal hybrid tree, we extract the hybrid sequence h_i below each MR production m_i. Using this process on the training N-M pairs, we can obtain a set of candidate hybrid sequences that can be associated with each MR production. The optimal hybrid tree generated by the Viterbi algorithm in this way is considered the "correct" hybrid tree for the N-M pair and is used as training data. While this does not provide hand-labeled training data, we believe the hybrid trees generated this way form a high quality training set as both the MR and NL are available when Viterbi decoding is performed, guaranteeing that the generated hybrid tree has the correct yield.

There exist several advantages of such a model over the simple generative model. First, this model allows features that specifically model the dependencies between neighboring hybrid sequences in the tree to be used. In addition, the model can efficiently capture long range dependencies between MR productions and hybrid sequences since each hybrid sequence is allowed to depend on the entire MR tree.

For features, we employ four types of simple features, as presented below. Note that the first three types of features are vertex features, and the last are edge features. Examples are given based on Figure 6.3. All the features are indicator functions, *i.e.*, a feature takes value 1 if a certain combination is present, and 0 otherwise. The last three features explicitly encode information from the tree structure of MR.

Hybrid sequence features : one hybrid sequence together with the associated MR production.

For example:

$$\begin{aligned} g_1 : \langle & \texttt{run through } \textsc{State}_1, \\ & \textsc{River} : \mathit{traverse}(\textsc{State}) \rangle ; \end{aligned}$$

Two-level hybrid sequence features : one hybrid sequence, its associated MR production, and the parent MR production.

For example:

$$\begin{aligned} g_2 : \langle & \texttt{run through } \textsc{State}_1, \\ & \textsc{River} : \mathit{traverse}(\textsc{State}), \\ & \textsc{River} : \mathit{exclude}(\textsc{River}_1, \textsc{River}_2) \rangle ; \end{aligned}$$

Three-level hybrid sequence features : one hybrid sequence, its associated

MR production, the parent MR production, and the grandparent MR production.

For example:

$$\begin{aligned} g_3 : \langle & \texttt{run through } \textsc{State}_1, \\ & \textsc{River} : \mathit{traverse}(\textsc{State}), \\ & \textsc{River} : \mathit{exclude}(\textsc{River}_1, \textsc{River}_2), \\ & \textsc{River} : \mathit{longest}(\textsc{River})\rangle\ ; \end{aligned}$$

Adjacent hybrid sequence features : two adjacent hybrid sequences, together with their associated MR productions.

For example:

$$\begin{aligned} f_1 : \langle & \texttt{run through } \textsc{State}_1, \\ & \textsc{River}_1 \texttt{ that does not } \textsc{River}_2, \\ & \textsc{River} : \mathit{traverse}(\textsc{State}), \\ & \textsc{River} : \mathit{exclude}(\textsc{River}_1, \textsc{River}_2)\rangle\ . \end{aligned}$$

For training, we use the feature forest model [MT08], which was originally designed as an efficient algorithm for solving maximum entropy models for data with complex structures. The model enables efficient training over packed trees that potentially represent exponential number of trees. The tree conditional random fields model can be effectively represented using the feature forest model. The model has also been successfully applied to the HPSG parsing task.

To train the model, we run the Viterbi algorithm on the trained generative model and perform convex optimization using the feature forest model. The generative model is trained using an EM algorithm with time complexity $O(mn^3d)$ per EM iteration, where m and n are respectively the maximum number of MR productions and NL words for each N-M pair, and d is the number of training instances. The time complexity of the Viterbi algorithm is also $O(mn^3d)$. For

training the feature forest, we use the Amis toolkit [MT02] which utilizes the GIS algorithm. The time complexity for each iteration of the GIS algorithm is $O(mk^2d)$, where K is the maximum number of candidate hybrid sequences associated with each MR production. Finally, the time complexity for generating a natural language sentence from a particular MR is $O(mk^2)$.

Chapter 7

Experiments and Discussions

7.1 Experiments for Semantic Parsing

Our evaluations were performed on two corpora, GEOQUERY and ROBOCUP. The GEOQUERY corpus contains MR defined by a Prolog-based language used in querying a database on U.S. geography. The ROBOCUP corpus contains MR defined by a coaching language used in a robot coaching competition. There are in total 880 and 300 instances for the two corpora respectively. Standard 10-fold cross validations were performed and the micro-averaged results are presented in this section. To make our system directly comparable to previous systems, all our experiments were based on identical training and test data splits of both corpora as reported in the experiments of Wong and Mooney [WM06].

7.1.1 Training Methodology

Given a training set, we first run a variant of IBM alignment model 1 [BPPM93] for 100 iterations, and then initialize the unigram model with the learned parameter values. This IBM model is a word-to-word alignment model that does not model word order, so we do not have to linearize the hierarchical MR structure. Given this initialization, we train the unigram model for 100 EM iterations and

use the learned parameters to initialize the bigram model which is trained for another 100 EM iterations. The mixgram model is simply an interpolation of the above two models. As for the reranking phase, we initialize the weight vector with the zero vector $\mathbf{0}$, and run the averaged perceptron algorithm for 10 iterations.

7.1.2 Evaluation Methodology

Following Wong [Won07] and other previous work, we report performance in terms of *Precision*, *Recall*, and F-score, which are defined as follows:

$$Precision = \frac{\text{Number of correct output MR trees}}{\text{Number of output MR trees}} \tag{7.1}$$

$$Recall = \frac{\text{Number of correct output MR trees}}{\text{Number of input NL sentences}} \tag{7.2}$$

$$F = \frac{2}{1/Precision + 1/Recall} \tag{7.3}$$

Again following Wong [Won07], we define the correct output MR structure as follows. For the GEOQUERY corpus, an MR structure is considered correct if and only if it retrieves identical results as the reference MR structure when both are issued as queries to the underlying Prolog database. For the ROBOCUP corpus, an MR structure is considered correct if and only if it has the same string representation as the reference MR structure, up to reordering of children of MR productions whose function symbols are commutative, such as *and*, *or*, etc.

7.1.3 Comparison Over Three Models

We evaluated the three models, with and without reranking. The results are presented in Table 7.1. Comparing the unigram model and the bigram model, we noticed that for both corpora, the unigram model in general achieves better recall while the bigram model achieves better precision. The mixgram model, as an interpolation of the above two models, achieves a much better F-measure on

	GEOQUERY (880)			ROBOCUP (300)		
	Prec.(%)	*Rec.*(%)	*F*(%)	*Prec.*(%)	*Rec.*(%)	*F*(%)
Unigram	81.3	77.1	79.1	71.1	**64.0**	67.4
Bigram	**89.0**	76.0	82.0	**82.4**	57.7	**67.8**
Mixgram	86.2	**81.8**	**84.0**	70.4	63.3	66.7
Unigram++	87.5	80.5	83.8	79.1	67.0	72.6
Bigram++	**93.2**	73.6	82.3	**88.4**	56.0	68.6
Mixgram++	89.3	**81.5**	**85.2**	82.5	**67.7**	**74.4**

Table 7.1: Performance comparison over three proposed semantic parsing models (*Prec.*:*Precision*, *Rec.*:*Recall*, *F*:*F*-score, ++ : with reranking)

the GEOQUERY corpus. However, it is shown to be less effective on the ROBOCUP corpus. We noticed that compared to the GEOQUERY corpus, the ROBOCUP corpus contains longer sentences, larger MR structures, and a significant amount of non-compositionality. These factors combine to present a challenging problem for parsing with the generative model. Interestingly, although the mixgram model fails to produce better best predictions for this corpus, we found that its top-k list contains a relatively larger number of correct predictions than the unigram model or the bigram model. This indicates the possibility of enhancing the performance with reranking.

The reranking approach is shown to be quite effective. We observe a consistent improvement in both precision and F-measure after employing the reranking phase for each model.

7.1.4 Comparison with Previous Models

Among all the previous models, SILT, WASP, and KRISP are directly comparable to our model. They required the same amount of supervision as our system and were evaluated on the same corpora.

We compare our model with these models in Table 7.2, where the performance scores for the previous systems are taken from [Won07]. For the GEOQUERY corpus, our model performs substantially better than all the three previous models, with a notable improvement in the recall score. In fact, if we look at the recall

scores alone, our best-performing model achieves a 6.7% and 9.8% absolute improvement over two other state-of-the-art models WASP and KRISP respectively. This indicates that overall, our model is able to handle over 25% of the inputs that could not be handled by previous systems. On the other hand, in terms of F-measure, we gain a 4.1% absolute improvement over KRISP, which leads to an error reduction rate of 22%. On the ROBOCUP corpus, our model's performance is also ranked the highest. Note that we are unable to perform statistical significance tests because the detailed performance for each fold of previously published research work is not available.

	GEOQUERY (880)			ROBOCUP (300)		
	Prec.(%)	*Rec.*(%)	*F*(%)	*Prec.*(%)	*Rec.*(%)	*F*(%)
SILT	89.0	54.1	67.3	83.9	50.7	63.2
WASP	87.2	74.8	80.5	**88.9**	61.9	73.0
KRISP	**93.3**	71.7	81.1	85.2	61.9	71.7
Mixgram++	89.3	**81.5**	**85.2**	82.5	**67.7**	**74.4**

Table 7.2: Performance comparison with other directly comparable semantic parsing systems

7.1.5 Performance on Different Languages

As a generic model that requires minimal assumptions on the natural language, our model is natural language independent and is able to handle various other natural languages than English. To validate this point, we evaluated our system on a subset of the GEOQUERY corpus consisting of 250 instances, with four different NL annotations.

As we can see from Table 7.3, our model is able to achieve performance comparable to WASP as reported by Wong [Won07].

Our model is generic, which requires no domain-dependent knowledge and should be applicable to a wide range of different domains. Like all research in this area, the ultimate goal is to scale to more complex, open-domain language understanding problems. In future, we would like to create a larger corpus in

	English			*Japanese*		
	Prec.(%)	*Rec.*(%)	*F*(%)	*Prec.*(%)	*Rec.*(%)	*F*(%)
WASP	95.42	70.00	80.76	91.98	74.40	**82.86**
Mixgram++	91.46	72.80	**81.07**	87.56	76.00	81.37
	Spanish			*Turkish*		
	Prec.(%)	*Rec.*(%)	*F*(%)	*Prec.*(%)	*Rec.*(%)	*F*(%)
WASP	91.99	72.40	81.03	96.96	62.40	75.93
Mixgram++	95.19	79.20	**86.46**	93.82	66.80	**78.04**

Table 7.3: Performance for semantic parsing on different natural languages for the GEOQUERY-250 corpus

another domain with multiple natural language annotations to further evaluate the scalability and portability of our approach.

7.2 Experiments for Natural Language Generation

In this section, we present the results of our systems when evaluated on the two standard benchmark corpora discussed in Section 7.1. Our task for this corpus is to generate NL sentences from the formal queries. The second corpus is ROBOCUP. This domain contains MRs which are instructions written in a formal language called CLANG. Our task for this domain is to generate NL sentences from the coaching advice written in CLANG [CFH+02].

Recall that the GEOQUERY corpus contains 880 instances, while the ROBOCUP corpus contains 300 instances. The average NL sentence length for the two corpora are 7.57 and 22.52 respectively. Following the evaluation methodology of Wong and Mooney [WM07a], we performed 4 runs of the standard 10-fold cross validation and report the averaged performance in this section using the standard automatic evaluation metric BLEU [PRWZ01] and NIST [Dod02].

The BLEU score is defined as follows:

$$\text{BLEU} = BP \times \exp \sum_{n=1}^{4} \frac{\log p_n}{4} \tag{7.4}$$

where BP is the brevity penalty that penalizes candidate translations that are shorter than the reference translation, and p_n is the n-gram precision of the candidate translations – in this task, it is the proportion of the overlapping n-grams between a candidate sentence and a reference over all n-grams in the candidate sentence.

The NIST score is defined as follows:

$$\text{NIST} = BP' \times \sum_{n=1}^{5} p'_n \tag{7.5}$$

Analogous to BLEU, the NIST metric also involves a brevity penalty BP' but is defined differently. The term p'_n denotes the weighted n-gram precision of candidate translations.

To standardize the evaluation tasks, we used the official evaluation script (version 11b) provided by http://www.nist.gov/. The BLEU and NIST scores of the WASP^{-1}++ system reported in this section are obtained from the published paper of Wong and Mooney [WM07a]. Note that to make our experimental results directly comparable to Wong and Mooney [WM07a], we used the identical training and test data splits for the 4 runs of 10-fold cross validation used by Wong and Mooney [WM07a] on both corpora.

Our system has the advantage of always producing an NL sentence given any input MR, even if there exist unseen MR productions in the input MR. We can achieve this by simply skipping those unseen MR productions during the generation process. However, in order to make a fair comparison against WASP^{-1}++, which can only generate NL sentences for 97% of the input MRs, we also do not generate any NL sentence in the case of observing an unseen MR production. All the evaluations discussed in this section follow this evaluation methodology, but we notice that empirically our system is able to achieve higher BLEU/NIST scores if we allow generation for those MRs that include unseen MR

productions.

7.2.1 Comparison Between the Two Models

	GEOQUERY (880)		ROBOCUP (300)	
	BLEU	NIST	BLEU	NIST
Direct inversion model	0.3973	5.5466	0.5468	6.6738
Tree CRF model	**0.5733**	**6.7459**	**0.6220**	**6.9845**

Table 7.4: Performance of automatic evaluation of both proposed language generation models (**bold** type indicates the best performing system).

We compare the performance of our two models in Table 7.4. From the table, we observe that the tree CRF model outperforms the direct inversion model on both domains. This validates our earlier belief that some long range dependencies are important for the generation task. In addition, while the direct inversion model performs reasonably well on the ROBOCUP domain, it performs substantially worse on the GEOQUERY domain where the sentence length is shorter. We note that the evaluation metrics are strongly correlated with the cumulative matching n-grams between the output and the reference sentence (n ranges from 1 to 4 for BLEU, and 1 to 5 for NIST). The direct inversion model fails to capture the transitional behavior from one phrase to another, which makes it more vulnerable to n-gram mismatch, especially when evaluated on the GEOQUERY corpus where phrase-to-phrase transitions are more frequent. On the other hand, the tree CRF model does not suffer from this problem, mainly due to its ability to model such dependencies between neighboring phrases. Sample outputs from the two models are shown in Figure 7.1.

7.2.2 Comparison with Previous Model

We also compare the performance of our tree CRF model against the previous state-of-the-art system WASP^{-1}++ in Table 7.5. Our tree CRF model achieves better performance on both corpora. We are unable to carry out statistical signif-

Reference:	`what is the largest state bordering texas`
Direct inversion model:	`what the largest states border texas`
Tree CRF model:	`what is the largest state that borders texas`
Reference:	`if DR2C7 is true then players 2 , 3 , 7 and 8 should pass to player 4`
Direct inversion model:	`if DR2C7 , then players 2 , 3 7 and 8 should ball to player 4`
Tree CRF model:	`if the condition DR2C7 is true then players 2 , 3 , 7 and 8 should pass to player 4`

Figure 7.1: Sample outputs from the two models, for GEOQUERY domain (top) and ROBOCUP domain (bottom) respectively.

	GEOQUERY (880)		ROBOCUP (300)	
	BLEU	NIST	BLEU	NIST
WASP^{-1}++	0.5370	6.4808	0.6022	6.8976
Tree CRF model	**0.5733**	**6.7459**	**0.6220**	**6.9845**

Table 7.5: Performance of automatic evaluation of tree CRF model and WASP^{-1}++ . (**bold** type indicates the best performing system)

icance tests since the detailed BLEU and NIST scores of the cross validation runs of WASP^{-1}++ as reported in the published paper of Wong and Mooney [WM07a] are not available.

The results confirm our earlier discussions: the dependencies between the generated NL words are important and need to be properly modeled. The WASP^{-1}++ system uses a log-linear model which incorporates two major techniques to attempt to model such dependencies. First, a back-off language model is used to capture dependencies at adjacent word level. Second, a technique that merges smaller translation rules into a single rigid rule is used to capture dependencies at phrase level [Won07]. In contrast, the proposed tree CRF model is able to explicitly and flexibly exploit phrase-level features that model dependencies between adjacent phrases. In fact, with the hybrid tree framework, the better treatment of the tree structure of MR enables us to model some crucial dependencies between the complete MR tree and generated NL phrases. We believe that

this property plays an important role in improving the quality of the generated sentences in terms of fluency, which is assessed by the evaluation metrics.

Furthermore, WASP^{-1}++ employs minimum error rate training [Och03] to directly optimize the evaluation metrics. We have not done so but still obtain better performance. In future, we plan to explore ways to directly optimize the evaluation metrics in our system.

7.2.3 Experiments on Different Languages

	English		*Japanese*	
	BLEU	NIST	BLEU	NIST
WASP^{-1}++	0.6035	5.7133	0.6585	4.6648
Tree CRF model	**0.6265**	**5.8907**	**0.6788**	**4.8486**
	Spanish		*Turkish*	
	BLEU	NIST	BLEU	NIST
WASP^{-1}++	0.6175	5.7293	0.4824	4.3283
Tree CRF model	**0.6382**	**5.8488**	**0.5096**	**4.5033**

Table 7.6: Performance on the GEOQUERY-250 corpus with 4 natural languages. (**bold** type indicates the best performing system)

Following the work of Wong and Mooney [WM07a], we also evaluated our system's performance on a subset of the GEOQUERY corpus with 250 instances, where sentences of 4 natural languages (English, Japanese, Spanish, and Turkish) are available. The evaluation results are shown in Table 7.6. Our tree CRF model achieves better performance on this task compared to WASP^{-1}++. We are again unable to conduct statistical significance tests for the same reason reported earlier.

Chapter 8

Future Work

In this chapter, we discuss some potential future research directions related to the hybrid tree framework presented in this thesis.

8.1 MR-Based Machine Translation

In this thesis, we have already demonstrated how the proposed hybrid tree framework can be utilized for semantic parsing and natural language generation. In other words, we can perform transformations from language to meaning representation, and vice versa. This motivates us to perform translation from one natural language to another through the use of meaning representation as the interlingua, as illustrated in Figure 8.1.

There are several potential advantages of such an MR-based machine translation approach over conventional machine translation approaches. Unlike conventional machine translation, where we need to train a separate machine translation system for each language pair (*e.g.*, English to Spanish and Spanish to English), in MR-based machine translation, only a separate model that translates between each individual natural language to the meaning representation is required, since the MR is shared amongst various different languages. This elegant property

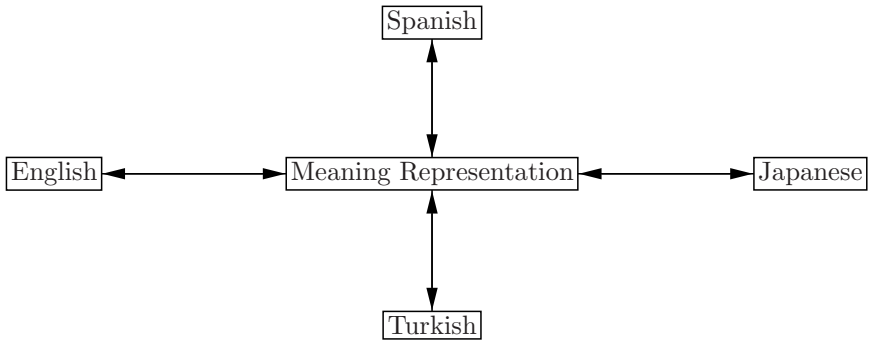


Figure 8.1: MR-based machine translation

saves substantial training effort. A conventional approach requires the building of $O(n^2)$ separate systems to perform pair-wise machine translation between each language pair, while the MR-based approach only requires training of $2n$ translation systems. Such an MR-based machine translation system is also relatively easier to incorporate new languages (introducing a new language just requires training 2 additional models) compared to conventional translation systems.

Though the MR-based machine translation has the advantages as discussed above, there also exist several potential difficulties. Perhaps the most important difficulty comes from the representation of meanings. Such an MR-based translation approach heavily rely on the meaning representation as the "interlingua" that bridges over different languages. However, it is still not quite clear what is a good meaning representation. On the other hand, even if we know what the best way of representing meaning is, it will take tremendous human efforts to annotate the meaning representations for large corpora. One possible solution to this problem can be to explore ways of performing a semi-supervised approach such as co-training [BM98]. In such an approach, only a few instances are fully annotated with the meaning representations while the majority are un-annotated.

8.2 Exploiting Linguistic Information

As we have discussed, one feature of the proposed generative model is that it requires minimal linguistic information. However, the proposed hybrid tree framework does not prevent us from exploiting such features.

We believe that certain linguistic information should be helpful in both the semantic parsing and language generation tasks. Consider the following two natural language sentences:

```
                which state has the highest population in the US ?
                which state has the largest population in the US ?
which state has higher population than all other states in the US ?
```

All the above three NL sentences convey the same semantics and should share the identical meaning representation. However, the proposed model is unable to capture the relationships among the word "`highest`", the word "`largest`", and the phrase "`higher ... than all other ...`". This is because the proposed generative model considers the generation of each individual NL word as an atomic generation step, where no linguistic information is involved.

Such linguistic information can in fact be quite useful in improving model effectiveness. For example, if one can make use of some external knowledge such as WordNet [Fel98] to encode the synonymous relationship between the word "`largest`" and the word "`highest`", much effort can be saved from the learning process and hence achieve improved performance. It is also possible to exploit some other types of linguistic information such as part-of-speech (POS) tags. Such additional information can be helpful in disambiguating words.

8.3 Support for Lambda Calculus

Though the hybrid tree framework is a generic framework that does not impose much restriction on the meaning representations, all the work presented in this thesis assumes the meaning representation is of a variable-free version and can be viewed as a tree structure. However, we believe it is possible to extend our current work to have wider range of support for other meaning representations, such as the lambda calculus as presented by Zettlemoyer and Collins [ZC05, ZC07].

Consider the λ-expression as given in Figure 2.9 of Section 2.2.5:

$$borders(utah, idaho) \tag{8.1}$$

which should map to the natural language sentence "`Utah borders Idaho`".

In fact, in a typed version of the lambda calculus [Bar84], each expression should be associated with a type. For example, the above expression should be of the type T (boolean type). Some other types include E (entity type) and more complicated compositional types such as $\langle \text{E}, \text{T} \rangle$.

With an associated types for each λ-expression, we can perform decomposition on a complete λ-expression. For example, the expression above can be written as an application of the following two sub-expressions:

$$\Big(\lambda x.borders(x, idaho) \qquad utah\Big)_{\text{Functional Application}} \tag{8.2}$$

Thus, the original meaning representation in λ-expression is decomposed into two subexpressions: a constant ($utah$) and another λ-expression which is $\lambda x.borders(x, idaho)$. It is not hard to observe that both subexpressions themselves convey complete semantics and can get mapped to natural language phrases. Continuing the decomposition of the λ-expressions and we eventually reach the parse-tree shown in Figure 8.2.

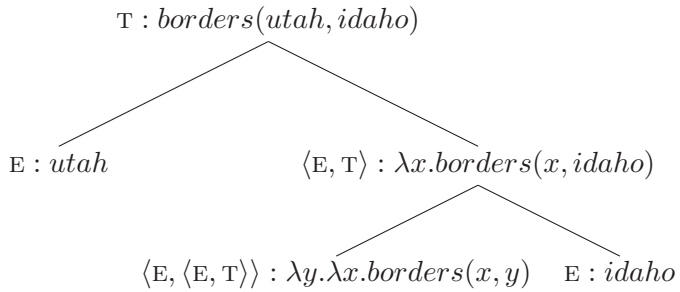


Figure 8.2: A tree structure of the MR

The resulting tree in Figure 8.2 can be considered the MR tree as used throughout this thesis. Now we can apply the generative models developed in this thesis accordingly, and perform semantic parsing as well as language generation with lambda calculus support. The correct hybrid tree for the above example is shown in Figure 8.3.

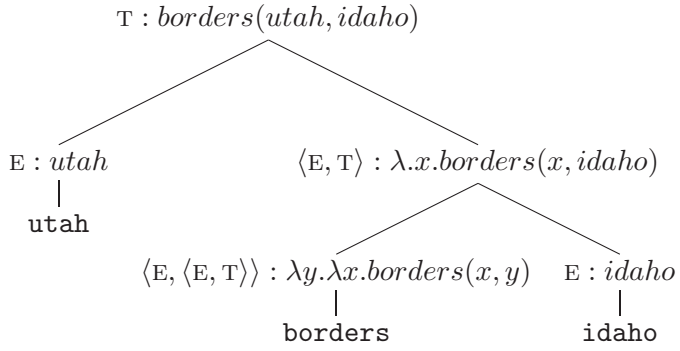


Figure 8.3: The hybrid tree that supports λ-calculus

However, there may exist different ways to decompose the same λ-expression. An alternative MR tree for the same λ-expression above is shown in Figure 8.4. With such an MR tree, the model will learn a different set of parameters. For some longer λ-expressions, there may exist even more possible different decompositions, and therefore many different MR trees. If the decomposition procedure is inconsistent over different training examples, one may not be able to obtain a set of parameters with reasonable quality.

Second, in the above approach, the semantic category associated with each

MR production is merely the type of that expression. While the type provides some information about the category of semantics conveyed by the λ-expression, it is in fact highly unspecific and are too general. This weakness may also negatively affect the proposed model's effectiveness.

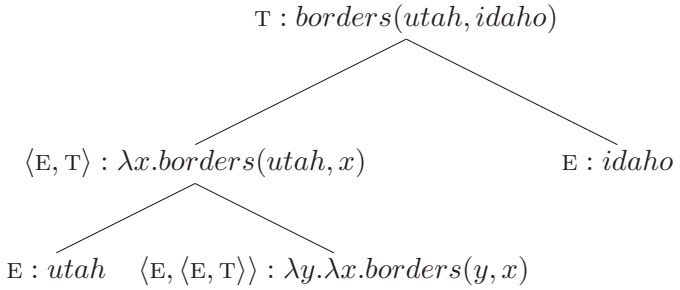


Figure 8.4: An alternative tree structure of the MR

Chapter 9

Conclusions

In this thesis, we presented a novel framework together with algorithms that bridge natural language with their underlying meaning representations. In particular, guided by certain theoretical principles related to language and semantics, we proposed a framework called *hybrid tree* in Chapter 3. As we have already surveyed in Chapter 2, one common approach for bridging natural languages and their meaning representations is to learn some form of probabilistic grammar which includes a list of lexical items that models the meanings of input words and also includes rules for combining lexical meanings to analyze complete sentences. This approach performs well but is constrained by the use of a single, learned grammar that contains a fixed set of lexical entries and productions. In practice, such a grammar may lack the rules required to correctly parse some of the new test examples.

On top of the proposed framework, we developed a generative model that present simultaneous generation processes for both natural languages and meaning representations in Chapter 4. The proposed generative model does not make use of an explicit grammar, but instead models the correspondence between sentences and their meanings with a generative process. This model is defined over the earlier introduced hybrid trees whose nodes include both natural language

words and meaning representation tokens. Inspired by the work of Collins [Col03], the generative model builds trees by recursively creating nodes at each level according to a Markov process. This implicit grammar representation leads to flexible learned models that generalize well. In practice, we observed that it can handle a wider range of test examples than previous approaches.

The generative model is learned from data that consists of NL sentences paired with their meaning representations. However, there is no explicit labeling of the correspondence between NL words and MR productions that is necessary for building the hybrid trees. This creates a challenging, hidden-variable learning problem that we address with the use of an inside-outside algorithm. Specifically, we develop a dynamic programming parsing algorithm that leads to $O(n^3m)$ time complexity for parameter estimation, where n is the sentence length and m is the number of MR productions appearing in the MR. With different assumptions during the parameter estimation phase, three models are presented and studied.

The proposed system can be used to perform two important tasks in the natural language processing field, namely semantic parsing (transforming natural language into formal symbolic representations of their meanings) and natural language generation (translating meaning representations into natural language sentences). In Chapter 6, we present methods for utilizing the proposed hybrid tree framework together with the generative model to perform semantic parsing and language generation.

For the semantic parsing task, in practice, we observe that the learned generative models are able to assign a high score to the correct meaning for input sentences, but that this correct meaning is not always the highest scoring option. To address this problem, we use a simple reranking approach to select the best parse from a k-best list of parses. As demonstrated through experiments in Chapter 7, this pipelined approach achieves state-of-the-art performance on two publicly available corpora, ROBOCUP and GEOQUERY. In particular, the flexible

generative model leads to notable improvements in recall, the total percentage of sentences that are correctly parsed.

We also believed that the better treatment of the MR structure can be helpful in the language generation task, which can be viewed as the inverse of semantic parsing. We first present a baseline model by directly "inverting" the semantic parsing model, where an NL sentence is generated word by word. We call this model the *direct inversion model*. This model is unable to capture some long range global dependencies over the entire NL sentence to be generated.

To tackle several weaknesses exhibited by the baseline model, we next introduce an alternative, *novel* model for generation. Motivated by conditional random fields [LMP01], a different parameterization of the same conditional model was presented that enables the model to encode some longer range dependencies amongst phrases and MRs. This novel model was referred to as the *tree* CRF *model*.

Evaluation results for both language generation models were presented in Chapter 7, through which we demonstrated that the phrase-based model performs better than the direct inversion model. We also compared the tree CRF model against the previous state-of-the-art model of Wong and Mooney [WM07a]. Comprehensive evaluations are presented, and evaluation results show that our proposed tree CRF model outperforms the previous model.

Appendices

Appendix A

Derivation for EM Formulas

let us denote the current model parameter as Ω, where $\Omega = \langle \phi, \theta, \rho \rangle$. We use M to denote the set of meaning representations, and N to denote the set of natural languages. We also use T to denote the set of hybrid trees associated with each N-M pair. Given this current parameter Ω, the joint likelihood is:

$$P(T, M, N; \Omega) \tag{A.1}$$

Consider

$$P(T|M, N; \Omega) = \frac{P(T, M, N; \Omega)}{P(M, N; \Omega)} \tag{A.2}$$

Equivalently,

$$\log P(M, N; \Omega) = \log P(T, M, N; \Omega) - \log P(T|M, N; \Omega) \tag{A.3}$$

let us use $l_c(\Omega; M, N, T)$ to denote the log-likelihood based on the complete data given a particular set of hybrid trees T, and use $l(\Omega; M, N)$ to denote the joint log-likelihood. The latter is the term that we want to maximize. We also use $l_h(\Omega; T|M, N)$ to denote the conditional log-likelihood for the set of hybrid trees, given the set of N-M pairs.

Thus, we have:

$$l(\Omega; M, N) = l_c(\Omega; T, M, N) - l_h(\Omega; T|M, N) \tag{A.4}$$

Now we would like to find the new parameter Ω' to maximize the term $l(\Omega'; M, N)$. Since T is the hidden variable, we take the expectation over the two terms appearing in the right hand side of Equation A.4, reaching the following:

$$l(\Omega'; M, N) = E_{P(T|M,N,\Omega)} l_c(\Omega'; T, M, N) - E_{P(T|M,N,\Omega)} l_h(\Omega'; T|M, N) \tag{A.5}$$

let us denote the two terms on the right hand side of Equation A.5 as $Q(\Omega', \Omega)$ and $R(\Omega', \Omega)$ respectively. In other words,

$$Q(\Omega', \Omega) = E_{P(T|M,N;\Omega)} l_c(\Omega'; T, M, N) \tag{A.6}$$
$$R(\Omega', \Omega) = E_{P(T|M,N;\Omega)} l_h(\Omega'; T|M, N) \tag{A.7}$$

Now, we would like to find the Ω' as follows:

$$\Omega' = \underset{\Omega'}{\operatorname{argmax}} \Big(Q(\Omega', \Omega) - R(\Omega', \Omega) \Big) \tag{A.8}$$

Note that

$$\begin{aligned}
R(\Omega, \Omega) - R(\Omega', \Omega) &= E_{P(T|M,N,\Theta)} l_h(\Omega; T|M, N) - E_{P(T|M,N,\Theta)} l_h(\Omega'; T|M, N) \\
&= E_{P(T|M,N,\Omega)} \Big(l_h(\Omega; T|M, N) - l_h(\Omega'; T|M, N) \Big) \\
&= E_{P(T|M,N;\Omega)} \Big(\log P(T|M, N; \Omega) - \log P(T|M, N; \Omega') \Big) \\
&= E_{P(T|M,N;\Omega)} \log \frac{P(T|M, N, \Omega)}{P(T|M, N, \Omega')} \\
&= \sum_T P(T|M, N; \Omega) \times \log \frac{P(T|M, N, \Omega)}{P(T|M, N, \Omega')} \\
&\geq 0
\end{aligned} \tag{A.9}$$

Note the last step is due to the famous Gibbs' inequality.

Therefore we have for any Ω':

$$R(\Omega', \Omega) - R(\Omega, \Omega) \leq 0 \tag{A.10}$$

Now let us consider $\Omega' = \Omega^*$ that maximizes $Q(\Omega', \Omega)$, in other words,

$$\Omega^* = \underset{\Omega'}{\operatorname{argmax}} Q(\Omega', \Omega) \tag{A.11}$$

Therefore,

$$Q(\Omega^*, \Omega) - Q(\Omega, \Omega) \geq 0 \tag{A.12}$$

Now we have:

$$\begin{aligned} l(\Omega^*; M, N) - l(\Omega; M, N) &= [Q(\Omega^*, \Omega) - Q(\Omega, \Omega)] - [R(\Omega^*, \Omega) - R(\Omega, \Omega)] \\ &\geq 0 \end{aligned} \tag{A.13}$$

In other words, the log-likelihood that we would like to optimize can be increased if we use this new parameter Ω^*. Thus, instead we find the Ω^* such that

$$\Omega^* = \underset{\Omega'}{\operatorname{argmax}} Q(\Omega', \Omega) \tag{A.14}$$

Now let us take a closer look at $Q(\Omega', \Omega)$, as follows:

$$\begin{aligned} & Q(\Omega', \Omega) \\ = \; & E_{P(T|M,N,;\Omega)} l_c(\Omega'; T, M, N) \\ = \; & E_{P(T|M,N,\Omega)} \log P(T, M, N; \Omega') \\ = \; & E_{P(T|M,N;\Omega)} \left(\sum_i \log P(T^i, M^i, N^i; \Omega') \right) \end{aligned} \tag{A.15}$$

The last step of Equation A.15 holds due to the fact that the joint probability for the whole corpus can be decomposed into individual independent probability terms, each is a joint probability for generating the hybrid tree that contains one N-M pair.

Since the N-M pairs are independent of each other, we can rewrite the expectation as follows:

$$\begin{aligned} & Q(\Omega', \Omega) \\ = & \sum_i E_{P(T^i|M^i,N^i;\Omega)} \log P(T^i, M^i, N^i; \Omega') \\ = & \sum_i E_{P(T^i|M^i,N^i;\Omega)} \Big(\log P(M^i; \Omega') + \log P(T^i, N^i|M^i; \Omega') \Big) \end{aligned} \tag{A.16}$$

Note that the hybrid tree T^i contains N^i and M^i, thus we can also write the above equation as follows:

$$\begin{aligned} & Q(\Omega', \Omega) \\ = & \sum_i E_{P(T^i|M^i,N^i;\Omega)} \Big(\log P(M^i; \Omega') + \log P(T^i|M^i; \Omega') \Big) \end{aligned} \tag{A.17}$$

The generative model introduced some independence assumptions. The generation of the MR depends on the MR model parameter ρ only, and the generation of the rest portion of the hybrid tree depends on the emission parameter θ and the pattern parameter ϕ only. Therefore, we have:

$$\begin{aligned} & (\rho^*, \theta^*, \phi^*) \\ = & \underset{(\rho', \theta', \phi')}{\operatorname{argmax}} \sum_i E_{P(T^i|M^i,N^i;\Omega)} \Big(\log P(M^i; \rho') + \log P(T^i|M^i; \theta', \phi') \Big) \end{aligned} \tag{A.18}$$

Therefore, we have:

$$\rho^* = \underset{\rho'}{\operatorname{argmax}} \sum_i E_{P(T^i|M^i,N^i;\Omega)} \log P(M^i; \rho') \tag{A.19}$$

and

$$(\theta^*, \phi^*) = \underset{(\theta', \phi')}{\operatorname{argmax}} \sum_i E_{P(T^i|M^i,N^i;\Omega)} \log P(T^i|M^i; \theta', \phi') \tag{A.20}$$

A.1 Updating MR Model Parameters

let us first look at the MR parameter ρ. Since M^i is independent of T^i for each possible i, we have:

$$\begin{aligned} & E_{P(T^i|M^i,N^i,\Omega)} \log P(T^i|M^i; \theta', \phi') \\ = & \sum_j P(T^i_j|M^i, N^i, \Omega) \times \log P(M^i; \rho') \\ = & \left(\sum_j P(T^i_j|M^i, N^i, \Omega) \right) \times \log P(M^i; \rho') \end{aligned} \tag{A.21}$$

Note that $P(T^i_j|M^i, N^i; \Omega)$ refers to the probability of the hybrid tree T^i_j given the N-M pair (M^i, N^i). The total probability mass for summing the probability of all the possible hybrid trees given an N-M pair is always 1. In other words,

$$\sum_j P(T^i_j|M^i, N^i, \Omega) = 1 \tag{A.22}$$

Therefore,

$$E_{P(T^i|M^i,N^i;\Omega)} \log P(T^i|M^i; \theta', \phi') = \log P(M^i; \rho') \tag{A.23}$$

Thus,

$$\rho^* = \underset{\rho'}{\operatorname{argmax}} \sum_i \log P(M^i; \rho') \tag{A.24}$$

let us consider a particular MR model parameter $\rho'(m_l|m_j, \arg_k)$. We would like to find the optimal $\rho'(m_l|m_j, \arg_k)$ subject to the constraint

$$\sum_{u} \rho'(m_u|m_j, \arg_k) = 1 \tag{A.25}$$

Assuming there are totally S possible values for m, we have:

$$\rho'(m_S|m_j, \arg_k) = 1 - \sum_{u \neq S} \rho'(m_u|m_j, \arg_k) \tag{A.26}$$

Thus, this term also involves the term $\rho'(m_l|m_j, \arg_k)$.

let us use $\text{Count}(m_u, m_j, \arg_k)$ to denote the number of times that m_u appears as the k-th child of m_j in the corpus. The objective function to be optimized in Equation A.24, excluding those terms that do not involve $\rho'(m_l|m_j, \arg_k)$ is as follows:

$$\begin{aligned} & \mathcal{L}(\rho'(m_l|m_j, \arg_k)) \\ = & \sum_{\text{Count}(m_l, m_j, \arg_k)} \Big(\log \rho'(m_l|m_j, \arg_k) + \log \rho'(m_S|m_j, \arg_k) \Big) \end{aligned} \tag{A.27}$$

This is a function of the parameter $\rho'(m_l|m_j, \arg_k)$. To optimize this function, we take the partial derivative, and make it equal to 0:

$$\begin{aligned} \frac{\partial \mathcal{L}(\rho'(m_l|m_j, \arg_k))}{\partial \rho'(m_l|m_j, \arg_k)} &= \sum_{\text{Count}(m_t, m_j, \arg_k)} \frac{1}{\rho'(m_l|m_j, \arg_k)} - \frac{1}{\rho'(m_S|m_j, \arg_k)} \\ &= \sum_{\text{Count}(m_t, m_j, \arg_k)} \frac{1}{\rho'(m_l|m_j, \arg_k)} \\ &- \sum_{\text{Count}(m_t, m_j, \arg_k)} \frac{1}{\rho'(m_S|m_j, \arg_k)} = 0 \end{aligned} \tag{A.28}$$

Now we have:

$$\frac{\text{Count}(m_l, m_j, \arg_k)}{\rho'(m_l|m_j, \arg_k)} = \frac{\text{Count}(m_S, m_j, \arg_k)}{\rho'(m_S|m_j, \arg_k)} \tag{A.29}$$

We therefore have:

$$\frac{\text{Count}(m_l, m_j, \arg_k)}{\rho'(m_l|m_j, \arg_k)} = \frac{\sum_u \text{Count}(m_u, m_j, \arg_k)}{\sum_u \rho'(m_u|m_j, \arg_k)} \tag{A.30}$$

We use $\text{Count}(m_j)$ to denote the number of times that m_j appeared in the corpus. We have:

$$\begin{aligned} \frac{\sum_u \text{Count}(m_u, m_j, \arg_k)}{\sum_u \rho'(m_u|m_j, \arg_k)} &= \sum_u \text{Count}(m_u, m_j, \arg_k) \\ &= \text{Count}(m_j) \end{aligned} \tag{A.31}$$

Thus, we have:

$$\rho'(m_l|m_j, \arg_k) = \frac{\text{Count}(m_l, m_j, \arg_k)}{\text{Count}(m_j)} \tag{A.32}$$

Note that this formula is independent of ρ, the old estimation. This means the formula needs only to be computed once, yielding the following formula:

$$\rho(m_l|m_j, \arg_k) = \frac{\text{Count}(m_l \text{ appeared in the corpus as the } k\text{-th child of } m_j)}{\text{Count}(m_j \text{ appeared in the corpus})} \tag{A.33}$$

A.2 Updating the Emission and Pattern Parameters

Now let us turn to the issue of updating the emission and pattern parameters.

$$(\theta^*, \phi^*) = \underset{(\theta', \phi')}{\text{argmax}} \sum_i E_{P(T^i|M^i, N^i; \theta, \phi)} \log P(T^i|M^i; \theta', \phi') \tag{A.34}$$

let us use T^i_j to denote the j-th possible hybrid tree for the i-th N-M pair, we have:

$$(\theta^*, \phi^*) = \underset{(\theta', \phi')}{\text{argmax}} \left(\sum_i \sum_j P(T_j^i | M^i, N^i; \theta, \phi) \times \log P(T_j^i | M^i; \theta', \phi') \right) \quad \text{(A.35)}$$

A.2.1 Optimization of Emission Parameters

Firstly we consider the optimization of $\theta'(t|m_v, \Lambda_k)$. Only those terms which are directly relevant to $\theta'(t|m_v, \Lambda_k)$ will potentially affect the optimization result. Therefore, only those hybrid trees that contain the tuple (t, m_v, Λ_k) will be retained. We have:

$$\begin{aligned} & \theta^*(t|m_v, \Lambda_k) \\ = & \underset{\theta'(t|m_v, \Lambda_k)}{\text{argmax}} \left(\sum_i \sum_{(t, m_v, \Lambda_k) \in T_j^i} P(T_j^i | M^i, N^i; \theta, \phi) \times \log \theta'(t|m_v, \Lambda_k) \right) \\ = & \underset{\theta'(t|m_v, \Lambda_k)}{\text{argmax}} \left(\sum_i \sum_{(t, m_v, \Lambda_k) \in T_j^i} P(T_j^i | M^i, N^i; \theta, \phi) \right) \times \log \theta'(t|m_v, \Lambda_k) \end{aligned} \quad \text{(A.36)}$$

where $(t, m_v, \Lambda_k) \in T_j^i$ means the tuple (t, m_v, Λ_k) appears in the hybrid tree T_j^i.

We now want to optimize the above objective function subject to the following constraint:

$$\sum_u \theta'(u|m_v, \Lambda_k) = 1 \quad \text{(A.37)}$$

Consider a particular NL word/semantic category S, we have:

$$\theta'(S|m_v, \Lambda_k) = 1 - \sum_{u \neq S} \theta'(u|m_v, \Lambda_k) \quad \text{(A.38)}$$

let us define $\text{Count}(T_j^i; t, m_v, \Lambda_k)$ as the number of times the tuple (t, m_v, Λ_k)

appears in T_j^i.

Now we define and compute the term:

$$\begin{aligned} & c^i(t, m_v, \Lambda_k; \Omega) \\ = & \sum_j \text{Count}(T_j^i; t, m_v, \Lambda_k) \times P(T_j^i | M^i, N^i; \Omega) \\ = & \sum_j \text{Count}(T_j^i; t, m_v, \Lambda_k) \times \frac{P(T_j^i, M^i, N^i; \Omega)}{P(M^i, N^i; \Omega)} \\ = & \frac{1}{P(M^i, N^i; \Omega)} \times \sum_j \text{Count}(T_j^i; t, m_v, \Lambda_k) \times P(T_j^i, M^i, N^i; \Omega) \end{aligned} \quad \text{(A.39)}$$

Note that the term $P(M^i, N^i; \Omega)$ is the joint probability for the N-M pair (M^i, N^i), which is exactly the inside probability for the N-M pair. We use $\beta(d^i)$ to denote this inside probability.

The remaining term of Equation A.39

$$\sum_j \text{Count}(T_j^i; t, m_v, \Lambda_k) \times P(T_j^i, M^i, N^i; \Omega) \quad \text{(A.40)}$$

is the total probability mass for all the hybrid trees that contain the tuple (t, m_v, Λ_k). In particular, if a hybrid tree contains k such tuples, the probability for that hybrid tree will be accumulated k times. This total probability mass can be computed efficiently with the help of inside and outside formulas defined in Chapter 3.

In summary, we have:

$$\begin{aligned} & c^i(t, m_v, \Lambda_k; \Omega) \\ = & \frac{1}{\beta(d^i)} \times \sum_{\substack{(t, m_v, \Lambda_k) \in h \\ h \in \textsc{Children}(m_v)}} \left(\alpha(d_v^i) \times P(h | m_v) \times \prod_{d_{v'}^i \in \textsc{Children}(h)} \beta(d_{v'}^i) \right) \end{aligned} \quad \text{(A.41)}$$

Note that the notation $(t, m_v, \Lambda_k) \in h$ means that the tuple (t, m_v, Λ_k) ap-

peared in the decomposed N-M pair h.

Denote

$$\begin{aligned}
& \mathcal{L}(\theta'(t|m_v,\Lambda_k)) \\
= & \left(\sum_i \sum_{(t,m_v,\Lambda_k)\in T_j^i} P(T_j^i|M^i,N^i;\theta,\phi)\right) \times \log\theta'(t|m_v,\Lambda_k) \\
= & \left(\sum_i \sum_j \text{Count}(T_j^i;t,m_v,\Lambda_k) \times P(T_j^i|M^i,N^i;\Omega)\right) \times \log\theta'(t|m_v,\Lambda_k) \\
= & \left(\sum_i c^i(t,m_v,\Lambda_k;\Omega)\right) \times \log\theta'(t|m_v,\Lambda_k) \qquad \text{(A.42)}
\end{aligned}$$

Take the partial derivative of the above term with respect to $\theta'(t|m_v,\Lambda_k)$:

$$\begin{aligned}
\frac{\partial\mathcal{L}(\theta'(t|m_v,\Lambda_k))}{\partial\theta'(t|m_v,\Lambda_k)} &= \left(\sum_i c^i(t,m_v,\Lambda_k;\Omega)\right) \times \frac{1}{\theta'(t|m_v,\Lambda_k)} \\
&- \left(\sum_i c^i(S,m_v,\Lambda_k;\Omega)\right) \times \frac{1}{\theta'(S|m_v,\Lambda_k)} \qquad \text{(A.43)}
\end{aligned}$$

Making the derivative to zero, we obtain

$$\frac{\sum_i c^i(t,m_v,\Lambda_k;\Omega)}{\theta'(t|m_v,\Lambda_k)} = \frac{\sum_i c^i(S,m_v,\Lambda_k;\Omega)}{\theta'(S|m_v,\Lambda_k)} \qquad \text{(A.44)}$$

Thus, we have

$$\frac{\sum_i c^i(t,m_v,\Lambda_k;\Omega)}{\theta'(t|m_v,\Lambda_k)} = \frac{\sum_{t'}\sum_i c^i(t',m_v,\Lambda_k;\Omega)}{\sum_{t'}\theta'(t'|m_v,\Lambda_k)} = \sum_{t'}\sum_i c^i(t',m_v,\Lambda_k;\Omega) \qquad \text{(A.45)}$$

Now we have the update equation for emission parameters:

$$\theta'(t|m_v,\Lambda_k) = \frac{\sum_i c^i(t,m_v,\Lambda_k;\Omega)}{\sum_{t'}\sum_i c^i(t',m_v,\Lambda_k;\Omega)} \qquad \text{(A.46)}$$

where the count function c is defined in Equation A.41.

A.2.2 Optimization of Pattern Parameters

Now let us look at the pattern parameter $\phi(r|m_v)$. Again, only those terms which are directly relevant to ϕ' will affect the optimization result. Therefore, we consider a particular $\phi'(r|m_v)$:

$$\begin{aligned} & \phi^*(r|m_v) \\ = & \operatorname*{argmax}_{\phi'(r|m_v)} \left(\sum_i \sum_{(r,m_v)\in T_j^i} P(T_j^i|M^i, N^i; \theta, \phi) \times \log \phi'(r|m_v) \right) \\ = & \operatorname*{argmax}_{\phi'(r|m_v)} \left(\sum_i \sum_{(r,m_v)\in T_j^i} P(T_j^i|M^i, N^i; \theta, \phi) \right) \times \log \phi'(r|m_v) \end{aligned} \quad \text{(A.47)}$$

We now want to optimize the value of $\phi'(r|m_v)$ subject to the following constraint:

$$\sum_u \phi'(u|m_v) = 1 \quad \text{(A.48)}$$

Consider a particular hybrid pattern S, we have:

$$\phi'(S|m_v) = 1 - \sum_{u\neq S} \phi'(u|m_v) \quad \text{(A.49)}$$

let us define $\text{Count}(T_j^i; r, m_v)$ as the number of times the tuple (r, m_v) appears in T_j^i.

Now we define and compute the term

$$
\begin{aligned}
& c^i(r, m_v; \Omega) \\
= & \sum_j \text{Count}(T_j^i; r, m_v) \times P(T_j^i | M^i, N^i; \Omega) \\
= & \sum_j \text{Count}(T_j^i; r, m_v) \times \frac{P(T_j^i, M^i, N^i; \Omega)}{P(M^i, N^i; \Omega)} \\
= & \frac{1}{P(M^i, N^i; \Omega)} \times \sum_j \text{Count}(T_j^i; r, m_v) \times P(T_j^i, M^i, N^i; \Omega) \qquad \text{(A.50)}
\end{aligned}
$$

As we have discussed earlier,

$$P(M^i, N^i; \Omega) = \beta(d^i) \qquad \text{(A.51)}$$

and

$$\sum_j \text{Count}(T_j^i; r, m_v) \times P(T_j^i, M^i, N^i; \Omega) \qquad \text{(A.52)}$$

is the total probability mass for all the hybrid trees that contain the tuple (r, m_v). In particular, if a hybrid tree contains k such tuples, the probability for that hybrid tree will be accumulated k times. This total probability mass can be computed efficiently with the help of inside and outside formulas defined in Chapter 3.

In summary, we have:

$$
\begin{aligned}
& c^i(r, m_v; \Omega) \\
= & \frac{1}{\beta(d^i)} \times \sum_{\substack{(r, m_v) \in h \\ h \in \textsc{Children}(m_v)}} \left(\alpha(d_v^i) \times P(h | m_v) \times \prod_{d_{v'}^i \in \textsc{Children}(h)} \beta(d_{v'}^i) \right) \qquad \text{(A.53)}
\end{aligned}
$$

Again, the notation $(r, m_v) \in h$ means the tuple (r, m_v) appears in the decomposed N-M pair h.

The objective function of $\phi'(r|m_v)$ to be optimized is:

$$\begin{aligned}\mathcal{L}\left(\phi'(r|m_v)\right) &= \left(\sum_i \sum_{(r,m_v)\in T_j^i} P(T_j^i|M^i,N^i;\theta,\phi)\right) \times \log \phi'(r|m_v) \\ &= \left(\sum_i c^i(r,m_v;\Omega)\right) \times \log \phi'(r|m_v) \end{aligned} \tag{A.54}$$

Take the partial derivative of the above term with respect to $\phi'(r|m_v)$:

$$\begin{aligned}\frac{\partial \mathcal{L}\left(\phi'(r|m_v)\right)}{\partial \phi'(r|m_v)} = &\left(\sum_i c^i(r,m_v;\Omega)\right) \times \frac{1}{\phi'(r|m_v)} \\ &- \left(\sum_i c^i(S,m_v;\Omega)\right) \times \frac{1}{\phi'(S|m_v)}\end{aligned} \tag{A.55}$$

Making the derivative to zero, we obtain

$$\frac{\sum_i c^i(r,m_v;\Omega)}{\phi'(r|m_v)} = \frac{\sum_i c^i(S,m_v;\Omega)}{\phi'(S|m_v)} \tag{A.56}$$

Thus, we have

$$\frac{\sum_i c^i(r,m_v;\Omega)}{\phi'(r|m_v)} = \frac{\sum_{r'}\sum_i c^i(r',m_v;\Omega)}{\sum_{r'}\phi'(r'|m_v)} = \sum_{r'}\sum_i c^i(r',m_v;\Omega) \tag{A.57}$$

Now we have the following update equation for the pattern parameters:

$$\phi'(r|m_v) = \frac{\sum_i c^i(r,m_v;\Omega)}{\sum_{r'}\sum_i c^i(r',m_v;\Omega)} \tag{A.58}$$

where the count function c is defined in Equation A.50.

Appendix B

Complete Formulas for Inside Probability Computation

In convention, we assume that the MR production m_x with context Λ_x. This MR production can have either zero, one or two children in its MR tree. If it has one child, the child is m_y with semantic category $\mathcal{M}_y$; if it has two children, the children are MR productions m_y and m_z, with semantic categories $\mathcal{M}_y$ and $\mathcal{M}_z$ respectively.

B.1 Decomposition with hybrid patterns

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x \\ \wedge \\ w_s^l \end{array}\right) = \sum_r \phi\left(r|m_x\right) \times p\left(\left\langle \begin{array}{c} m_x \\ \wedge \\ w_s^l \end{array}, r \right\rangle\right) \tag{B.1}$$

where $\langle \mathcal{T}, r \rangle$ is defined as the total possible ways of constructing the next level hybrid sequence directly under the production m_x with the rule pattern r.

If m_x has no child, we have:

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathbf{w} \right\rangle\right) = \mathcal{P}_\beta \left(\overset{m_x(\texttt{BEGIN})}{\underset{w_s^l}{\frown}} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.2}$$

If m_x has one child, we have:

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathcal{Y} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ | \\ \mathcal{M}_y \\ \frown \\ w_s^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \tag{B.3}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathcal{Y}\mathbf{w} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagdown \\ \mathcal{M}_y \quad w_*^l \\ \frown \\ w_s^* \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.4}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathbf{w}\mathcal{Y} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagdown \\ w_s^* \quad \mathcal{M}_y \\ \quad \frown \\ \quad w_*^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \tag{B.5}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad | \quad \diagdown \\ w_s^* \quad \mathcal{M}_y \quad w_*^l \\ \quad \frown \\ \quad w+ \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.6}$$

If m_x has two children, we have:

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathcal{Y}\mathcal{Z} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagdown \\ \mathcal{M}_y \quad \mathcal{M}_z \\ \frown \quad \frown \\ w_s^* \quad w_*^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_z) \tag{B.7}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathcal{Z}\mathcal{Y} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \mathcal{M}_z \quad \mathcal{M}_y \\ \triangle \quad \triangle \\ w_s^* \quad w_*^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \tag{B.8}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathbf{w}\mathcal{Y}\mathcal{Z} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ w_s^* \quad \mathcal{M}_y \quad \mathcal{M}_z \\ \quad\quad \triangle \quad \triangle \\ \quad\quad w+ \quad w_*^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_z) \tag{B.9}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathbf{w}\mathcal{Z}\mathcal{Y} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ w_s^* \quad \mathcal{M}_z \quad \mathcal{M}_y \\ \quad\quad \triangle \quad \triangle \\ \quad\quad w+ \quad w_*^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \tag{B.10}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \\ \triangle \quad\quad \triangle \\ w_s^* \quad\quad w_*^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_z) \tag{B.11}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathcal{Z}\mathbf{w}\mathcal{Y} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \\ \triangle \quad\quad \triangle \\ w_s^* \quad\quad w_*^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \tag{B.12}$$

$$p\left(\left\langle \begin{array}{c} m_x \\ \overline{\qquad} \\ w_s^l \end{array}, m \rightarrow \mathcal{Y}\mathcal{Z}\mathbf{w} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad | \quad \diagdown \\ \mathcal{M}_y \quad \mathcal{M}_z \quad w_*^l \\ \overline{\qquad} \quad \overline{\qquad} \\ w_s^* \quad w+ \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.13}$$

$$p\left(\left\langle \begin{array}{c} m_x \\ \overline{\qquad} \\ w_s^l \end{array}, m \rightarrow \mathcal{Z}\mathcal{Y}\mathbf{w} \right\rangle\right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad | \quad \diagdown \\ \mathcal{M}_z \quad \mathcal{M}_y \quad w_*^l \\ \overline{\qquad} \quad \overline{\qquad} \\ w_s^* \quad w+ \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.14}$$

$$p\left(\left\langle \begin{array}{c} m_x \\ \overline{\qquad} \\ w_s^l \end{array}, m \rightarrow \mathbf{w}\mathcal{Y}\mathcal{Z}\mathbf{w} \right\rangle\right)$$
$$= \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagup \quad \diagdown \quad \diagdown \\ w_s^* \quad \mathcal{M}_y \quad \mathcal{M}_z \quad w_*^l \\ \quad \overline{\qquad} \quad \overline{\qquad} \\ \quad w+ \quad w+ \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.15}$$

$$p\left(\left\langle \begin{array}{c} m_x \\ \overline{\qquad} \\ w_s^l \end{array}, m \rightarrow \mathbf{w}\mathcal{Z}\mathcal{Y}\mathbf{w} \right\rangle\right)$$
$$= \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagup \quad \diagdown \quad \diagdown \\ w_s^* \quad \mathcal{M}_z \quad \mathcal{M}_y \quad w_*^l \\ \quad \overline{\qquad} \quad \overline{\qquad} \\ \quad w+ \quad w+ \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.16}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z} \right\rangle\right)$$

$$= \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \hline w_s^* \quad \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \\ \quad\;\; w+ \quad\;\; w_*^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_z) \tag{B.17}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y} \right\rangle\right)$$

$$= \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \hline w_s^* \quad \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \\ \quad\;\; w+ \quad\;\; w_*^l \end{array} \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \tag{B.18}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w} \right\rangle\right)$$

$$= \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \hline \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \quad w_*^l \\ \quad w_s^* \quad\;\; w+ \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.19}$$

$$p\left(\left\langle \overset{m_x}{\underset{w_s^l}{\frown}}, m \rightarrow \mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w} \right\rangle\right)$$

$$= \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \hline \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \quad w_*^l \\ \quad w_s^* \quad\;\; w+ \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.20}$$

$$p\left(\left\langle \begin{array}{c} m_x \\ \widehat{} \\ w_s^l \end{array}, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w} \right\rangle\right)$$

$$= \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagup \quad | \quad \diagdown \quad \diagdown \\ w_s^* \quad \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \quad w_*^l \\ \quad \widehat{} \qquad\qquad \widehat{} \\ \quad w+ \qquad\qquad w+ \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.21}$$

$$p\left(\left\langle \begin{array}{c} m_x \\ \widehat{} \\ w_s^l \end{array}, m \rightarrow \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w} \right\rangle\right)$$

$$= \mathcal{P}_\beta \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagup \quad | \quad \diagdown \quad \diagdown \\ w_s^* \quad \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \quad w_*^l \\ \quad \widehat{} \qquad\qquad \widehat{} \\ \quad w+ \qquad\qquad w+ \end{array} \right) \times \theta(\texttt{END}|m_x, w_l) \tag{B.22}$$

B.2 Computation for aggregated hybrid trees

First we consider the case where m_x has two children.

For the case where the hybrid pattern is $m \rightarrow \mathcal{Y}\mathcal{Z}$ and $m \rightarrow \mathcal{Z}\mathcal{Y}$:

$$\mathcal{P}_\beta \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ \mathcal{M}_y \quad \mathcal{M}_z \\ \widehat{} \qquad \widehat{} \\ w_s^* \qquad w_*^l \end{array} \right) = \sum_{s+1 \le i \le l} \left[\mathcal{P}_\beta \left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_y \\ \widehat{} \\ w_s^{i-1} \end{array} \right) \times \mathcal{P}_\beta \left(\begin{array}{c} m_x(\mathcal{M}_y) \\ | \\ \mathcal{M}_z \\ \widehat{} \\ w_i^l \end{array} \right) \right] \tag{B.23}$$

$$\mathcal{P}_\beta \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ \mathcal{M}_z \quad \mathcal{M}_y \\ \widehat{} \qquad \widehat{} \\ w_s^* \qquad w_*^l \end{array} \right) = \sum_{s+1 \le i \le l} \left[\mathcal{P}_\beta \left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_z \\ \widehat{} \\ w_s^{i-1} \end{array} \right) \times \mathcal{P}_\beta \left(\begin{array}{c} m_x(\mathcal{M}_z) \\ | \\ \mathcal{M}_y \\ \widehat{} \\ w_i^l \end{array} \right) \right] \tag{B.24}$$

For the case where the hybrid pattern is $m \rightarrow \mathbf{w}\mathcal{Y}\mathcal{Z}$ and $m \rightarrow \mathbf{w}\mathcal{Z}\mathcal{Y}$:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \quad \mathcal{M}_z \\ \quad\; w+ \quad\; w_*^l \end{array}\right) = \sum_{s+2\le i\le l}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \\ \quad\; w_*^{i-1} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \\ w_i^l \end{array}\right)\right] \tag{B.25}$$

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_z \quad \mathcal{M}_y \\ \quad\; w+ \quad\; w_*^l \end{array}\right) = \sum_{s+2\le i\le l}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_z \\ \quad\; w_*^{i-1} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ \mathcal{M}_y \\ w_i^l \end{array}\right)\right] \tag{B.26}$$

For the case where the hybrid pattern is $m \to \mathcal{Y}\mathbf{w}\mathcal{Z}$ and $m \to \mathcal{Z}\mathbf{w}\mathcal{Y}$:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \\ \;\; w_s^* \quad\;\; w_*^l \end{array}\right) = \sum_{s+1\le i\le l-1}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \\ w_s^{i-1} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ w_i^* \quad \mathcal{M}_z \\ \quad\; w_*^l \end{array}\right)\right] \tag{B.27}$$

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \\ \;\; w_s^* \quad\;\; w_*^l \end{array}\right) = \sum_{s+1\le i\le l-1}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \\ w_s^{i-1} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ w_i^* \quad \mathcal{M}_y \\ \quad\; w_*^l \end{array}\right)\right] \tag{B.28}$$

For the case where the hybrid pattern is $m \to \mathcal{Y}\mathcal{Z}\mathbf{w}$ and $m \to \mathcal{Z}\mathcal{Y}\mathbf{w}$:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad \mathcal{M}_z \quad w_*^l \\ \underset{w_s^*}{\frown} \quad \underset{w+}{\frown} \end{array}\right) = \sum_{s+1\le i\le l-1}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \\ \underset{w_s^{i-1}}{\frown} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \quad w_*^l \\ \underset{w_i^*}{\frown} \end{array}\right)\right] \tag{B.29}$$

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad \mathcal{M}_y \quad w_*^l \\ \underset{w_s^*}{\frown} \quad \underset{w+}{\frown} \end{array}\right) = \sum_{s+1\le i\le l-1}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \\ \underset{w_s^{i-1}}{\frown} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ \mathcal{M}_y \quad w_*^l \\ \underset{w_i^*}{\frown} \end{array}\right)\right] \tag{B.30}$$

For the case where the hybrid pattern is $m \to \mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}$ and $m \to \mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w}$:

$$\begin{aligned} &\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \quad w_*^l \\ \underset{w_s^*}{\frown} \quad\quad \underset{w+}{\frown} \end{array}\right) \\ &= \sum_{s+2\le i\le l-1}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w_*^{i-1} \\ \underset{w_s^*}{\frown} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(w_{i-1}) \\ \mathcal{M}_z \quad w_*^l \\ \underset{w_i^*}{\frown} \end{array}\right)\right] \end{aligned} \tag{B.31}$$

$$\begin{aligned} &\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \quad w_*^l \\ \underset{w_s^*}{\frown} \quad\quad \underset{w+}{\frown} \end{array}\right) \\ &= \sum_{s+2\le i\le l-1}\left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad w_*^{i-1} \\ \underset{w_s^*}{\frown} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(w_{i-1}) \\ \mathcal{M}_y \quad w_*^l \\ \underset{w_i^*}{\frown} \end{array}\right)\right] \end{aligned} \tag{B.32}$$

For the case where the hybrid pattern is $m \to \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}$ and $m \to \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y}$:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \\ \quad\; w+ \quad\quad\; w_*^l \end{array}\right)$$

$$= \sum_{s+2\le i\le l-1} \left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \\ \quad\; w_*^{i-1} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ w_i^* \quad \mathcal{M}_z \\ \quad\; w_*^l \end{array}\right) \right] \tag{B.33}$$

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \\ \quad\; w+ \quad\quad\; w_*^l \end{array}\right)$$

$$= \sum_{s+2\le i\le l-1} \left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_z \\ \quad\; w_*^{i-1} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ w_i^* \quad \mathcal{M}_y \\ \quad\; w_*^l \end{array}\right) \right] \tag{B.34}$$

For the case where the hybrid pattern is $m \to \mathbf{w}\mathcal{Y}\mathcal{Z}\mathbf{w}$ and $m \to \mathbf{w}\mathcal{Z}\mathcal{Y}\mathbf{w}$:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \quad \mathcal{M}_z \quad w_*^l \\ \quad\; w+ \quad\; w+ \end{array}\right)$$

$$= \sum_{s+2\le i\le l-1} \left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \\ \quad\; w_*^{i-1} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \quad w_*^l \\ \quad w_i^* \end{array}\right) \right] \tag{B.35}$$

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagup \quad \diagup \quad \diagup \\ w_s^* \quad \mathcal{M}_z \quad \mathcal{M}_y \quad w_*^l \\ \quad\quad \underline{\quad w+ \quad} \quad \underline{\quad w+ \quad} \end{array}\right)$$

$$= \sum_{s+2\le i\le l-1} \left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagup \\ w_s^* \quad \mathcal{M}_z \\ \quad\quad \underline{\quad w_*^{i-1} \quad} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ \diagup \quad \diagup \\ \mathcal{M}_y \quad w_*^l \\ \underline{\quad w_i^* \quad} \end{array}\right) \right] \tag{B.36}$$

For the case where the hybrid pattern is $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}$ and $m \rightarrow \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w}$:

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagup \quad \diagup \quad \diagup \quad \diagup \\ w_s^* \quad \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \quad w_*^l \\ \quad\quad \underline{\quad w+ \quad} \quad\quad \underline{\quad w+ \quad} \end{array}\right)$$

$$= \sum_{s+3\le i\le l-1} \left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagup \quad \diagup \\ w_s^* \quad \mathcal{M}_y \quad w_*^{i-1} \\ \quad\quad \underline{\quad w+ \quad} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(w_{i-1}) \\ \diagup \quad \diagup \\ \mathcal{M}_z \quad w_*^l \\ \underline{\quad w_i^* \quad} \end{array}\right) \right] \tag{B.37}$$

$$\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagup \quad \diagup \quad \diagup \quad \diagup \\ w_s^* \quad \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \quad w_*^l \\ \quad\quad \underline{\quad w+ \quad} \quad\quad \underline{\quad w+ \quad} \end{array}\right)$$

$$= \sum_{s+3\le i\le l-1} \left[\mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagup \quad \diagup \\ w_s^* \quad \mathcal{M}_z \quad w_*^{i-1} \\ \quad\quad \underline{\quad w+ \quad} \end{array}\right) \times \mathcal{P}_\beta\left(\begin{array}{c} m_x(w_{i-1}) \\ \diagup \quad \diagup \\ \mathcal{M}_y \quad w_*^l \\ \underline{\quad w_i^* \quad} \end{array}\right) \right] \tag{B.38}$$

Now we should consider how to deal with $\mathcal{P}_\beta$ that involve one child only. The child is either the first child or the second. For illustrative purpose, we consider the case of first child ($\mathcal{M}_y$) only.

It is important to note that these formulas are also directly applicable to the case where m_x has only one child.

For the case where the hybrid pattern is $m \rightarrow \mathcal{Y}$:

$$\mathcal{P}_\beta \left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_y \\ \widehat{\quad} \\ w_s^l \end{array} \right) = \theta(\mathcal{M}_y | m_x, \Lambda_k) \times \rho(m_y | m_x, \arg_1) \times \mathcal{P}_\beta \left(\begin{array}{c} m_y \\ \widehat{\quad} \\ w_s^l \end{array} \right) \tag{B.39}$$

where m_y is the child MR production of m_x with the semantic category $\mathcal{M}_y$.

For the case where the hybrid pattern is $m \rightarrow \mathcal{Y}\mathbf{w}$:

$$\begin{aligned} \mathcal{P}_\beta \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ \mathcal{M}_y \quad w_*^l \\ \widehat{\quad} \\ w_s^* \end{array} \right) = \mathcal{P}_\beta \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ \mathcal{M}_y \quad w_*^{l-1} \\ \widehat{\quad} \\ w_s^* \end{array} \right) \times \theta(w_l | m_x, w_{l-1}) \\ + \mathcal{P}_\beta \left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_y \\ \widehat{\quad} \\ w_s^{l-1} \end{array} \right) \times \theta(w_l | m_x, \mathcal{M}_y) \end{aligned} \tag{B.40}$$

For the case where the hybrid pattern is $m \rightarrow \mathbf{w}\mathcal{Y}$:

$$\mathcal{P}_\beta \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ w_s^* \quad \mathcal{M}_y \\ \quad \widehat{\quad} \\ \quad w_*^l \end{array} \right) = \theta(w_s | m_x, \Lambda_k) \times \left[\mathcal{P}_\beta \left(\begin{array}{c} m_x(w_s) \\ \diagup \quad \diagdown \\ w_{s+1}^* \quad \mathcal{M}_y \\ \quad \widehat{\quad} \\ \quad w_*^l \end{array} \right) + \mathcal{P}_\beta \left(\begin{array}{c} m_x(w_s) \\ | \\ \mathcal{M}_y \\ \widehat{\quad} \\ w_{s+1}^l \end{array} \right) \right] \tag{B.41}$$

For the case where the hybrid pattern is $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}$:

$$\mathcal{P}_\beta\left(\begin{array}{ccc} & m_x(\Lambda_k) & \\ w_s^* & \mathcal{M}_y & w_*^l \\ & \overline{\quad w+\quad} & \end{array}\right) = \theta(w_s|m_x,\Lambda_k)$$

$$\times\left[\mathcal{P}_\beta\left(\begin{array}{ccc} & m_x(w_s) & \\ w_{s+1}^* & \mathcal{M}_y & w_*^l \\ & \overline{\quad w+\quad} & \end{array}\right) + \mathcal{P}_\beta\left(\begin{array}{ccc} & m_x(w_s) & \\ & \mathcal{M}_y \quad w_*^l & \\ & \overline{\quad w_{s+1}^*\quad} & \end{array}\right)\right] \tag{B.42}$$

Appendix C

Complete Formulas for Outside Probability Computation

In convention, we assume that the MR production m_x with context Λ_x. This MR production can have either zero, one or two children in its MR tree. If it has one child, the child is m_y with semantic category $\mathcal{M}_y$; if it has two children, the children are MR productions m_y and m_z, with semantic categories $\mathcal{M}_y$ and $\mathcal{M}_z$ respectively.

C.1 Decomposition with hybrid patterns

First of all, consider the given training N-M pair consists of the complete MR tree rooted by m_a, and the complete sentence $w_1 \dots w_n$:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x \\ \overbrace{\quad\quad} \\ w_1^n \end{array} \right) = \left\{ \begin{array}{ll} 1 & m_x \equiv m_a \\ 0 & \text{otherwise} \end{array} \right. \tag{C.1}$$

In general, we have:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x \\ \wedge \\ w_s^l \end{array} \right) \times \phi(r|m_x) \twoheadrightarrow q \left(\left\langle \begin{array}{c} m_x \\ \wedge \\ w_s^l \end{array}, r \right\rangle \right) \tag{C.2}$$

The q term can in turn be computed as follows.

If m_x has one child only, we have:

$$q \left(\left\langle \begin{array}{c} m_x \\ \wedge \\ w_s^l \end{array}, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w} \right\rangle \right) \times \theta(\texttt{END}|m_x, w_l) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \; | \; \diagdown \\ w_s^* \quad \mathcal{M}_y \quad w_*^l \\ \quad \wedge \\ \quad w+ \end{array} \right) \tag{C.3}$$

$$q \left(\left\langle \begin{array}{c} m_x \\ \wedge \\ w_s^l \end{array}, m \rightarrow \mathbf{w}\mathcal{Y} \right\rangle \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagdown \\ w_s^* \quad \mathcal{M}_y \\ \quad \wedge \\ \quad w_*^l \end{array} \right) \tag{C.4}$$

$$q \left(\left\langle \begin{array}{c} m_x \\ \wedge \\ w_s^l \end{array}, m \rightarrow \mathcal{Y}\mathbf{w} \right\rangle \right) \times \theta(\texttt{END}|m_x, w_l) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \diagup \quad \diagdown \\ \mathcal{M}_y \quad w_*^l \\ \wedge \\ w_s^* \end{array} \right) \tag{C.5}$$

$$q \left(\left\langle \begin{array}{c} m_x \\ \wedge \\ w_s^l \end{array}, m \rightarrow \mathcal{Y} \right\rangle \right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ | \\ \mathcal{M}_y \\ \wedge \\ w_s^l \end{array} \right) \tag{C.6}$$

If m_x has two children, we have:

$$q\left(\left\langle \begin{array}{c} m_x \\ \overset{\frown}{w_s^l}, m \rightarrow \mathcal{Y}\mathcal{Z} \end{array} \right\rangle\right) \times \theta(\texttt{END}|m_x, \mathcal{M}_z) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \mathcal{M}_y \quad \mathcal{M}_z \\ \overset{\frown}{w_s^*} \quad \overset{\frown}{w_*^l} \end{array} \right) \tag{C.7}$$

$$q\left(\left\langle \begin{array}{c} m_x \\ \overset{\frown}{w_s^l}, m \rightarrow \mathcal{Z}\mathcal{Y} \end{array} \right\rangle\right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \mathcal{M}_z \quad \mathcal{M}_y \\ \overset{\frown}{w_s^*} \quad \overset{\frown}{w_*^l} \end{array} \right) \tag{C.8}$$

$$\begin{aligned} q\left(\left\langle \begin{array}{c} m_x \\ \overset{\frown}{w_s^l}, m \rightarrow \mathbf{w}\mathcal{Y}\mathcal{Z} \end{array} \right\rangle\right) \times \theta(\texttt{END}|m_x, \mathcal{M}_z) \\ \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ w_s^* \quad \mathcal{M}_y \quad \mathcal{M}_z \\ \quad\quad \overset{\frown}{w+} \quad \overset{\frown}{w_*^l} \end{array} \right) \end{aligned} \tag{C.9}$$

$$\begin{aligned} q\left(\left\langle \begin{array}{c} m_x \\ \overset{\frown}{w_s^l}, m \rightarrow \mathbf{w}\mathcal{Z}\mathcal{Y} \end{array} \right\rangle\right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y) \\ \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ w_s^* \quad \mathcal{M}_z \quad \mathcal{M}_y \\ \quad\quad \overset{\frown}{w+} \quad \overset{\frown}{w_*^l} \end{array} \right) \end{aligned} \tag{C.10}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z} \right\rangle\right) \times \theta(\texttt{END}|m_x, \mathcal{M}_z)$$

$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{ccc} & m_x(\texttt{BEGIN}) & \\ \mathcal{M}_y & w+ & \mathcal{M}_z \\ \overset{}{\underset{w_s^*}{\triangle}} & & \overset{}{\underset{w_*^l}{\triangle}} \end{array} \right) \tag{C.11}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathcal{Z}\mathbf{w}\mathcal{Y} \right\rangle\right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y)$$

$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{ccc} & m_x(\texttt{BEGIN}) & \\ \mathcal{M}_z & w+ & \mathcal{M}_y \\ \overset{}{\underset{w_s^*}{\triangle}} & & \overset{}{\underset{w_*^l}{\triangle}} \end{array} \right) \tag{C.12}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathcal{Y}\mathcal{Z}\mathbf{w} \right\rangle\right) \times \theta(\texttt{END}|m_x, w_l)$$

$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{ccc} & m_x(\texttt{BEGIN}) & \\ \mathcal{M}_y & \mathcal{M}_z & w_*^l \\ \overset{}{\underset{w_s^*}{\triangle}} & \overset{}{\underset{w+}{\triangle}} & \end{array} \right) \tag{C.13}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathcal{Z}\mathcal{Y}\mathbf{w} \right\rangle\right) \times \theta(\texttt{END}|m_x, w_l)$$

$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{ccc} & m_x(\texttt{BEGIN}) & \\ \mathcal{M}_z & \mathcal{M}_y & w_*^l \\ \overset{}{\underset{w_s^*}{\triangle}} & \overset{}{\underset{w+}{\triangle}} & \end{array} \right) \tag{C.14}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z} \right\rangle\right) \times \theta(\texttt{END}|m_x, \mathcal{M}_z)$$
$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \hline w_s^* \quad \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \\ \quad\quad\; w+ \quad\quad\; w_*^l \end{array} \right) \tag{C.15}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y} \right\rangle\right) \times \theta(\texttt{END}|m_x, \mathcal{M}_y)$$
$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \hline w_s^* \quad \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \\ \quad\quad\; w+ \quad\quad\; w_*^l \end{array} \right) \tag{C.16}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w} \right\rangle\right) \times \theta(\texttt{END}|m_x, m_l)$$
$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \hline \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \quad w_*^l \\ w_s^* \quad\quad\; w_*^l \end{array} \right) \tag{C.17}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w} \right\rangle\right) \times \theta(\texttt{END}|m_x, m_l)$$
$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ \hline \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \quad w_*^l \\ w_s^* \quad\quad\; w_*^l \end{array} \right) \tag{C.18}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathbf{w}\mathcal{Y}\mathcal{Z}\mathbf{w} \right\rangle\right) \times \theta(\texttt{END}|m_x, m_l)$$

$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ w_s^* \quad \mathcal{M}_y \quad \mathcal{M}_z \quad w_*^l \\ \quad\; w+ \quad\; w+ \end{array} \right) \tag{C.19}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathbf{w}\mathcal{Z}\mathcal{Y}\mathbf{w} \right\rangle\right) \times \theta(\texttt{END}|m_x, m_l)$$

$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ w_s^* \quad \mathcal{M}_z \quad \mathcal{M}_y \quad w_*^l \\ \quad\; w+ \quad\; w+ \end{array} \right) \tag{C.20}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w} \right\rangle\right) \times \theta(\texttt{END}|m_x, m_l)$$

$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ w_s^* \quad \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \quad w_*^l \\ \quad\; w+ \qquad\quad\; w+ \end{array} \right) \tag{C.21}$$

$$q\left(\left\langle \overset{m_x}{\underset{w_s^l}{\triangle}}, m \rightarrow \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w} \right\rangle\right) \times \theta(\texttt{END}|m_x, m_l)$$

$$\twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\texttt{BEGIN}) \\ w_s^* \quad \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \quad w_*^l \\ \quad\; w+ \qquad\quad\; w+ \end{array} \right) \tag{C.22}$$

C.2 Computation for aggregated hybrid trees

First we consider the case where m_x has two children.

For the case of expecting a pattern of the form $m \to \mathcal{Y}\mathcal{Z}$, for all $s+1 \leq i \leq l$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup\quad\diagdown \\ \mathcal{M}_y \quad \mathcal{M}_z \\ \overbrace{\quad}^{} \quad \overbrace{\quad}^{} \\ w_s^* \quad w_*^l \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ | \\ \mathcal{M}_z \\ \overbrace{\quad}^{} \\ w_i^l \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_y \\ \overbrace{\quad}^{} \\ w_s^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_y \\ \overbrace{\quad}^{} \\ w_s^{i-1} \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ | \\ \mathcal{M}_z \\ \overbrace{\quad}^{} \\ w_i^l \end{array}\right) \end{array} \right. \tag{C.23}$$

For the case of expecting a pattern of the form $m \to \mathcal{Z}\mathcal{Y}$, for all $s+1 \leq i \leq l$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup\quad\diagdown \\ \mathcal{M}_z \quad \mathcal{M}_y \\ \overbrace{\quad}^{} \quad \overbrace{\quad}^{} \\ w_s^* \quad w_*^l \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ | \\ \mathcal{M}_y \\ \overbrace{\quad}^{} \\ w_i^l \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_z \\ \overbrace{\quad}^{} \\ w_s^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_z \\ \overbrace{\quad}^{} \\ w_s^{i-1} \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ | \\ \mathcal{M}_y \\ \overbrace{\quad}^{} \\ w_i^l \end{array}\right) \end{array} \right. \tag{C.24}$$

For the case of expecting a pattern of the form $m \to \mathbf{w}\mathcal{Y}\mathcal{Z}$, for all $s+2 \leq i \leq l$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{ccc} & m_x(\Lambda_k) & \\ w_s^* & \mathcal{M}_y & \mathcal{M}_z \\ & w+ & w_*^l \end{array}\right) \rightarrow \left\{\begin{array}{ll} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \\ w_i^l \end{array}\right) & \rightarrow \quad \mathcal{P}_\alpha\left(\begin{array}{cc} m_x(\Lambda_k) & \\ w_s^* & \mathcal{M}_y \\ & w_*^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{cc} m_x(\Lambda_k) & \\ w_s^* & \mathcal{M}_y \\ & w_*^{i-1} \end{array}\right) & \rightarrow \quad \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \\ w_i^l \end{array}\right) \end{array}\right. \tag{C.25}$$

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Z}\mathcal{Y}$, for all $s+2 \leq i \leq l$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{ccc} & m_x(\Lambda_k) & \\ w_s^* & \mathcal{M}_z & \mathcal{M}_y \\ & w+ & w_*^l \end{array}\right) \rightarrow \left\{\begin{array}{ll} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ \mathcal{M}_y \\ w_i^l \end{array}\right) & \rightarrow \quad \mathcal{P}_\alpha\left(\begin{array}{cc} m_x(\Lambda_k) & \\ w_s^* & \mathcal{M}_z \\ & w_*^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{cc} m_x(\Lambda_k) & \\ w_s^* & \mathcal{M}_z \\ & w_*^{i-1} \end{array}\right) & \rightarrow \quad \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ \mathcal{M}_y \\ w_i^l \end{array}\right) \end{array}\right. \tag{C.26}$$

For the case of expecting a pattern of the form $m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z}$, for any $s+2 \leq i \leq l$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \\ \widehat{w_s^*} \qquad\qquad \widehat{w_*^l} \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(m_{i-1}) \\ | \\ \mathcal{M}_z \\ \widehat{w_i^l} \end{array}\right) & \rightarrowtriangle & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w_*^{i-1} \\ \widehat{w_s^*} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w_*^{i-1} \\ \widehat{w_s^*} \end{array}\right) & \rightarrowtriangle & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(m_{i-1}) \\ | \\ \mathcal{M}_z \\ \widehat{w_i^l} \end{array}\right) \end{array} \right. \tag{C.27}$$

For the case of expecting a pattern of the form $m \rightarrow \mathcal{Z}\mathbf{w}\mathcal{Y}$, for any $s+2 \leq i \leq l$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \\ \widehat{w_s^*} \qquad\qquad \widehat{w_*^l} \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(m_{i-1}) \\ | \\ \mathcal{M}_y \\ \widehat{w_i^l} \end{array}\right) & \rightarrowtriangle & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad w_*^{i-1} \\ \widehat{w_s^*} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad w_*^{i-1} \\ \widehat{w_s^*} \end{array}\right) & \rightarrowtriangle & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(m_{i-1}) \\ | \\ \mathcal{M}_y \\ \widehat{w_i^l} \end{array}\right) \end{array} \right. \tag{C.28}$$

For the case of expecting a pattern of the form $m \rightarrow \mathcal{Y}\mathcal{Z}\mathbf{w}$, for any $s+1 \leq i \leq l-1$,we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad \mathcal{M}_z \quad w_*^l \\ w_s^* \quad w+ \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \quad w_*^l \\ w_i^* \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \\ w_s^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \\ w_s^{i-1} \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ \mathcal{M}_z \quad w_*^l \\ w_i^* \end{array}\right) \end{array} \right. \tag{C.29}$$

For the case of expecting a pattern of the form $m \rightarrow \mathcal{Z}\mathcal{Y}\mathbf{w}$, for any $s+1 \leq i \leq l-1$,we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad \mathcal{M}_y \quad w_*^l \\ w_s^* \quad w+ \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ \mathcal{M}_y \quad w_*^l \\ w_i^* \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \\ w_s^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \\ w_s^{i-1} \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ \mathcal{M}_y \quad w_*^l \\ w_i^* \end{array}\right) \end{array} \right. \tag{C.30}$$

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}$, for any $s+2 \leq i \leq l-1$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \\ \quad\; w+ \qquad\quad w_*^l \end{array}\right) \times \left\{\begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ w_i^* \quad \mathcal{M}_z \\ \qquad\; w_*^l \end{array}\right) & \twoheadrightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \\ \qquad\; w_*^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \\ \qquad\; w_*^{i-1} \end{array}\right) & \twoheadrightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_y) \\ w_i^* \quad \mathcal{M}_z \\ \qquad\; w_*^l \end{array}\right) \end{array}\right. \tag{C.31}$$

For the case of expecting a pattern of the form $m \to \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y}$, for any $s+2 \leq i \leq l-1$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \\ \quad\; w+ \qquad\quad w_*^l \end{array}\right) \times \left\{\begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ w_i^* \quad \mathcal{M}_y \\ \qquad\; w_*^l \end{array}\right) & \twoheadrightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_z \\ \qquad\; w_*^{i-1} \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_z \\ \qquad\; w_*^{i-1} \end{array}\right) & \twoheadrightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\mathcal{M}_z) \\ w_i^* \quad \mathcal{M}_y \\ \qquad\; w_*^l \end{array}\right) \end{array}\right. \tag{C.32}$$

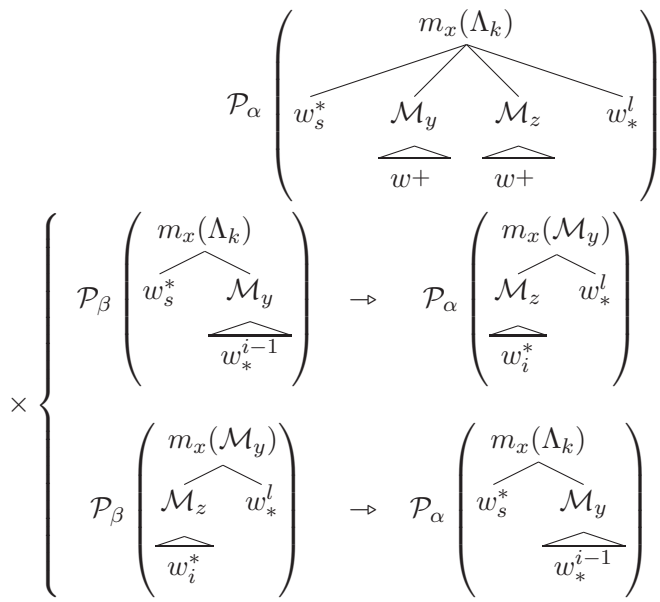
For the case of expecting a pattern of the form $m \to \mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}$, for any $s+2 \leq i \leq l-1$,we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w+ \quad \mathcal{M}_z \quad w_*^l \\ \underline{\quad\quad} \qquad \underline{\quad\quad} \\ w_s^* \qquad w+ \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(w_{i-1}) \\ \mathcal{M}_z \quad w_*^l \\ \underline{\quad\quad} \\ w_i^* \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w_*^{i-1} \\ \underline{\quad\quad} \\ w_s^* \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_y \quad w_*^{i-1} \\ \underline{\quad\quad} \\ w_s^* \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(w_{i-1}) \\ \mathcal{M}_z \quad w_*^l \\ \underline{\quad\quad} \\ w_i^* \end{array}\right) \end{array} \right. \tag{C.33}$$

For the case of expecting a pattern of the form $m \rightarrow \mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w}$, for any $s+2 \leq i \leq l-1$,we have:

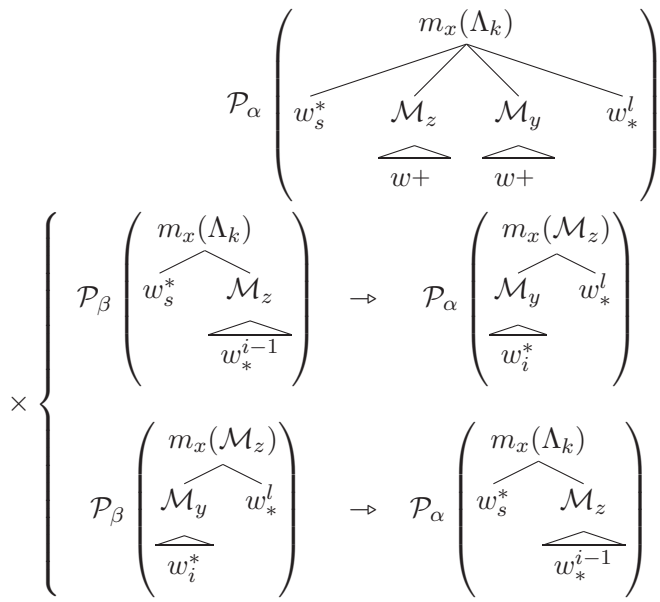
$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad w+ \quad \mathcal{M}_y \quad w_*^l \\ \underline{\quad\quad} \qquad \underline{\quad\quad} \\ w_s^* \qquad w+ \end{array}\right) \times \left\{ \begin{array}{lcl} \mathcal{P}_\beta\left(\begin{array}{c} m_x(w_{i-1}) \\ \mathcal{M}_y \quad w_*^l \\ \underline{\quad\quad} \\ w_i^* \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad w_*^{i-1} \\ \underline{\quad\quad} \\ w_s^* \end{array}\right) \\ \mathcal{P}_\beta\left(\begin{array}{c} m_x(\Lambda_k) \\ \mathcal{M}_z \quad w_*^{i-1} \\ \underline{\quad\quad} \\ w_s^* \end{array}\right) & \rightarrow & \mathcal{P}_\alpha\left(\begin{array}{c} m_x(w_{i-1}) \\ \mathcal{M}_y \quad w_*^l \\ \underline{\quad\quad} \\ w_i^* \end{array}\right) \end{array} \right. \tag{C.34}$$

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Y}\mathcal{Z}\mathbf{w}$, for any $s+2 \leq i \leq l-1$, we have:



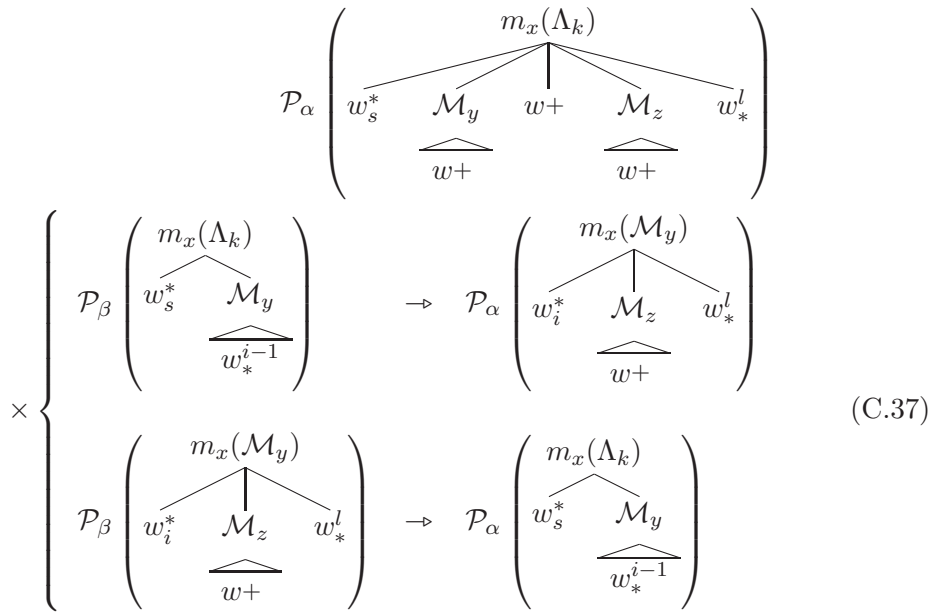
(C.35)

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Z}\mathcal{Y}\mathbf{w}$, for any $s+2 \leq i \leq l-1$, we have:



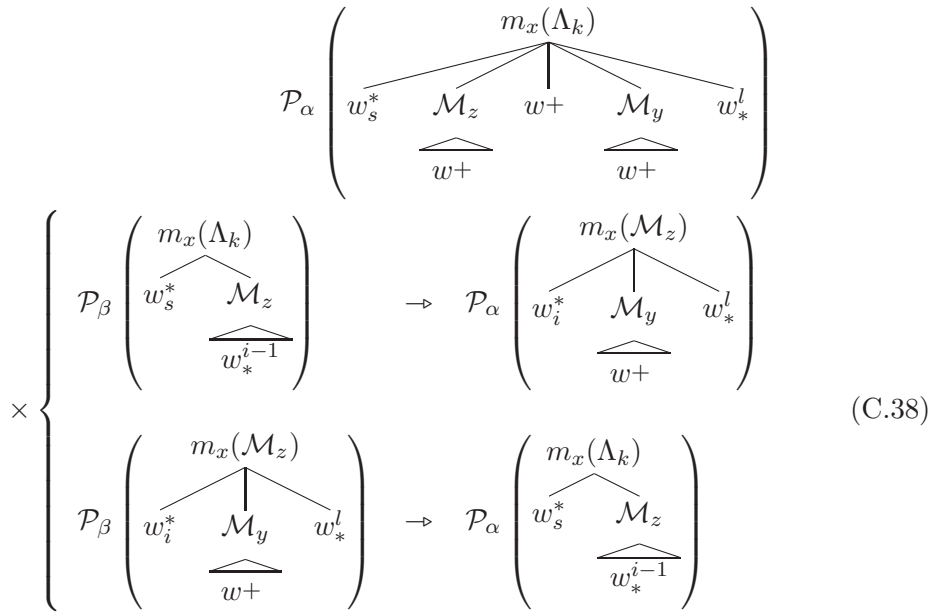
(C.36)

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}\mathbf{w}$, for any $s+2 \leq i \leq l-2$, we have:



(C.37)

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Z}\mathbf{w}\mathcal{Y}\mathbf{w}$, for any $s+2 \leq i \leq l-2$, we have:



(C.38)

Now we should consider how to deal with $\mathcal{P}_\alpha$ that involve one child only. The child is either the first child or the second. For illustrative purpose, we consider the case of first child ($\mathcal{M}_y$) only.

It is important to note that these formulas are also directly applicable to the case where m_x has only one child.

For the case of expecting a pattern of the form $m \rightarrow \mathcal{Y}$, we have:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_y \\ \frown \\ w_s^l \end{array} \right) \times \theta(\mathcal{M}_y | m_x, \Lambda_k) \times \rho(m_y | m_x, \arg_1) \twoheadrightarrow \mathcal{P}_\alpha \left(\begin{array}{c} m_y \\ \frown \\ w_s^l \end{array} \right) \tag{C.39}$$

where m_y is the child MR production of m_x with the semantic category $\mathcal{M}_y$.

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Y}$, we have:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ w_s^* \quad \mathcal{M}_y \\ \quad \frown \\ \quad w_*^l \end{array} \right) \times \theta(w_s | m_x, \Lambda_k) \quad \twoheadrightarrow \quad \left\{ \begin{array}{l} \mathcal{P}_\alpha \left(\begin{array}{c} m_x(w_s) \\ \diagup \quad \diagdown \\ w_{s+1}^* \quad \mathcal{M}_y \\ \quad \frown \\ \quad w_*^l \end{array} \right) \\ \\ \mathcal{P}_\alpha \left(\begin{array}{c} m_x(w_s) \\ | \\ \mathcal{M}_y \\ \frown \\ w_{s+1}^l \end{array} \right) \end{array} \right. \tag{C.40}$$

For the case of expecting a pattern of the form $m \rightarrow \mathcal{Y}\mathbf{w}$, we have:

$$\mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ \mathcal{M}_y \quad w_*^l \\ \frown \\ w_s^* \end{array} \right) \times \left\{ \begin{array}{lcl} \theta(w_l | m_x, w_{l-1}) & \twoheadrightarrow & \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ \diagup \quad \diagdown \\ \mathcal{M}_y \quad w_*^l \\ \frown \\ w_s^* \end{array} \right) \\ \\ \theta(w_l | m_x, \mathcal{M}_y) & \twoheadrightarrow & \mathcal{P}_\alpha \left(\begin{array}{c} m_x(\Lambda_k) \\ | \\ \mathcal{M}_y \\ \frown \\ w_s^{l-1} \end{array} \right) \end{array} \right. \tag{C.41}$$

For the case of expecting a pattern of the form $m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}$, we have:

$$\mathcal{P}_\alpha\left(\begin{array}{c} m_x(\Lambda_k) \\ w_s^* \quad \mathcal{M}_y \quad w_*^l \\ w+ \end{array}\right) \times \theta(w_s|m_x,\Lambda_k) \quad \rightarrow \quad \left\{\begin{array}{l} \mathcal{P}_\alpha\left(\begin{array}{c} m_x(w_s) \\ w_{s+1}^* \quad \mathcal{M}_y \quad w_*^l \\ w+ \end{array}\right) \\ \mathcal{P}_\alpha\left(\begin{array}{c} m_x(w_s) \\ \mathcal{M}_y \quad w_*^l \\ w_{s+1}^* \end{array}\right) \end{array}\right. \tag{C.42}$$

Bibliography

[ABD+01] James F. Allen, Donna K. Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. Toward conversational human-computer interaction. *AI magazine*, 22(4):27–37, 2001. [cited at p. 4]

[Bak79] James K. Baker. Trainable grammars for speech recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*, pages 547–550, Boston, MA, June 1979. [cited at p. 41]

[Bar84] Hendrik Pieter Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North Holland, New York, 1984. [cited at p. 108]

[Bat96] John A. Bateman. *KPML Development Environment: multilingual linguistic resource development and sentence generation*. GMD Forschungszentrum Informationstechnik, 1996. [cited at p. 20]

[BM98] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM New York, NY, USA, 1998. [cited at p. 106]

[BPPM93] Peter E Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993. [cited at p. 14, 95]

[CFH+02] Mao Chen, Ehsan Foroughi, Fredrik Heintz, ZhanXiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Itsuki Noda, Oliver Obst,

Pat Riley, Timo Steffens, Yi Wang, and Xiang Yin. *RoboCup soccer server users manual (for Soccer Server Version 7.07 and later).* The RoboCup Federation, 2002. [cited at p. 99]

[CK05] Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005. [cited at p. 17, 84]

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (Second Edition).* MIT Press, Cambridge, MA, 2001. [cited at p. 88]

[Col01] Michael Collins. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 489–496, Philadelphia, Pennsylvania, 2001. Association for Computational Linguistics. [cited at p. 81]

[Col02] Michael Collins. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8, Philadelphia, PA, 2002. Association for Computational Linguistics. [cited at p. 81]

[Col03] Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637, 2003. [cited at p. 17, 33, 40, 112]

[CST00] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-based Learning Methods.* Cambridge University Press, 2000. [cited at p. 15]

[DLR77] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the *em* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977. [cited at p. 41, 49]

[Dod02] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the 2nd International*

Conference on Human Language Technology Research (HLT 2002), pages 138–145, 2002. [cited at p. 99]

[Ear70] Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970. [cited at p. 15]

[EPR+96] Mark Epstein, Kishore A. Papineni, Salim Roukos, Todd R. Ward, and Stephen D. Pietra. Statistical natural language understanding using hidden clumpings. In *Proceedings of the 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1996)*, pages 176–179, Washington, DC, USA, 1996. IEEE Computer Society. [cited at p. 10]

[ER96] Michael Elhadad and Jacques Robin. An overview of SURGE: A reusable comprehensive syntactic realization component. In *8th International Natural Language Generation Workshop. Demonstrations and Posters*, pages 1–4, 1996. [cited at p. 20]

[Fel98] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998. [cited at p. 107]

[GJ02] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002. [cited at p. 1]

[GM05] Ruifang Ge and Raymond J. Mooney. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL 2005)*, pages 9–16, 2005. [cited at p. 10, 16]

[GM06] Ruifang Ge and Raymond J. Mooney. Discriminative reranking for semantic parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, pages 263–270, 2006. [cited at p. 10, 17]

[Hir92] Lynette Hirschman. Multi-site data collection for a spoken language corpus. In *Proceedings of the workshop on Speech and Natural Language, Human Language Technology Conference (HLT 1991)*, pages 7–14, Harriman, New York, 1992. Association for Computational Linguistics. [cited at p. 2, 10]

[HRWL83] Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat. *Building expert systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983. [cited at p. 1]

[JM08] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (Second Edition).* Prentice Hall, 2008. [cited at p. 7]

[Kat87] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401, 1987. [cited at p. 39]

[KM06] Rohit J. Kate and Raymond J. Mooney. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, pages 913–920, 2006. [cited at p. 10, 15]

[KM07a] Rohit J. Kate and Raymond J. Mooney. Learning language semantics from ambiguous supervision. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007)*, pages 895–900. AAAI Press, 2007. [cited at p. 10, 16]

[KM07b] Rohit J. Kate and Raymond J. Mooney. Semi-supervised learning for semantic parsing using support vector machines. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-07)*, pages 81–84, 2007. [cited at p. 10, 16]

[KMK01] Min-Yen Kan, Kathleen R. McKeown, and Judith L. Klavans. Applying natural language generation to indicative summarization. In *Proceedings of the 8th European workshop on Natural Language Generation (EWNLG 2001)*, pages 1–9, Toulouse, France, 2001. Association for Computational Linguistics. [cited at p. 4]

[Koe04] Philipp Koehn. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Conference of the*

Association for Machine Translation in the Americas (AMTA 2004), pages 115–124, 2004. [cited at p. 20]

[KOM03] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL 2003)*, pages 48–54, Edmonton, Canada, 2003. Association for Computational Linguistics. [cited at p. 21]

[KWM05] Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 1062–1068. AAAI Press, 2005. [cited at p. 7, 8, 10]

[LK98] Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL 1998)*, pages 704–710, Montreal, Quebec, Canada, 1998. Association for Computational Linguistics. [cited at p. 20]

[LMP01] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 282–289, Williams College, Williamstown, MA, 2001. Morgan Kaufmann Publishers Inc. [cited at p. 90, 91, 113]

[LNL09] Wei Lu, Hwee Tou Ng, and Wee Sun Lee. Natural language generation with tree conditional random fields. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, Suntec, Singapore, 2009. Association for Computational Linguistics. [cited at p. 5]

[LNLZ08] Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 783–792, Waikiki, Honolulu, HI, 2008. Association for Computational Linguistics. [cited at p. 5, 91]

[LR97] Benoit Lavoie and Owen Rambow. A fast and portable realizer for text generation systems. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP 1997)*, pages 265–268, Washington, DC, 1997. Association for Computational Linguistics. [cited at p. 20]

[LS68] P. M. Lewis, II and R. E. Stearns. Syntax-directed transduction. *J. ACM*, 15(3):465–488, 1968. [cited at p. 13]

[LSST+02] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002. [cited at p. 15]

[Man01] Inderjeet Mani. *Automatic Summarization*. John Benjamins Publishing Company, 2001. [cited at p. 3]

[MON01] Klaus Macherey, Franz J. Och, and Hermann Ney. Natural language understanding using statistical machine translation. In *7th European Conference on Speech Communication and Technology (EuroSpeech 2001)*, pages 2205–2208, Aalborg, Denmark, 2001. [cited at p. 10]

[Moo04] Raymond J. Mooney. Learning semantic parsers: An important but understudied problem. In *AAAI 2004 Spring Symposium on Language Learning: An Interdisciplinary Perspective*, pages 39–44. AAAI Press, 2004. [cited at p. 2]

[MT02] Yusuke Miyao and Jun'ichi Tsujii. Maximum entropy estimation for feature forests. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT 2002)*, pages 292–297, San Diego, California, 2002. Morgan Kaufmann Publishers Inc. [cited at p. 94]

[MT08] Yusuke Miyao and Jun'ichi Tsujii. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80, 2008. [cited at p. 93]

[NH04] Srini Narayanan and Sanda Harabagiu. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics (COLING 2004)*, pages 693–701, Geneva, Switzerland, 2004. Association for Computational Linguistics. [cited at p. 3]

[NL96] Hwee Tou Ng and Hian Beng Lee. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics (ACL1996)*, pages 40–47, 1996. [cited at p. 1]

[NM92] Eric H. Nyberg III and Teruko Mitamura. The KANT system: fast, accurate, high-quality translation in practical domains. In *Proceedings of the 14th conference on Computational linguistics - Volume 3 (COLING 1992)*, pages 1069–1073, Nantes, France, 1992. Association for Computational Linguistics. [cited at p. 3]

[Och03] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 160–167, 2003. [cited at p. 21, 103]

[ON03] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003. [cited at p. 14]

[OR00] Alice H. Oh and Alexander I. Rudnicky. Stochastic language generation for spoken dialogue systems. In *Proceedings of Workshop on Conversational systems, the Language Technology Joint Conference on Applied Natural Language Processing and the North American Chapter of the Association of Computational Linguistics (ANLP/NAACL 2000)*, pages 27–32, Seattle, Washington, 2000. Association for Computational Linguistics. [cited at p. 4]

[Par82] Barbara H. Partee. Compositionality. In Fred Landman and Frank Veltman, editors, *Varieties of Formal Semantics: Proceedings of the Fourth Amsterdam Colloquium*, volume 3, pages 281–311, September 1982. [cited at p. 27]

[PERW97] Stephen D. Pietra, Mark Epstein, Salim Roukos, and Todd R. Ward. Fertility models for statistical natural language understanding. In *Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL 1997)*, pages 168–173, Madrid, Spain, 1997. Association for Computational Linguistics. [cited at p. 10]

[PRW97] Kishore A. Papineni, Salim. Roukos, and Todd R. Ward. Feature-based language understanding. In *5th European Conference on Speech Communication and Technology (EuroSpeech 1997)*, pages 1435–1438, Rhodes, Greece, September 1997. [cited at p. 2, 10]

[PRWZ01] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 311–318, 2001. [cited at p. 99]

[Rat00] Adwait Ratnaparkhi. Trainable methods for surface natural language generation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics Conference (NAACL 2000)*, pages 194–201, Seattle, Washington, 2000. Morgan Kaufmann Publishers Inc. [cited at p. 20]

[RKPJ00] Stefan Riezler, Jonas Kuhn, Detlef Prescher, and Mark Johnson. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000)*, pages 480–487, Morristown, NJ, USA, 2000. Association for Computational Linguistics. [cited at p. 14]

[Ste96] Mark Steedman. *Surface Structure and Interpretation.* The MIT Press, Cambridge, MA, 1996. [cited at p. 18]

[Ste00] Mark Steedman. *The Syntactic Process.* The MIT Press, Cambridge, MA, 2000. [cited at p. 18]

[Tho77] Henry Thompson. Strategy and tactics: A model for language production. In *Papers from the 13th Regional Meeting of the Chicago Linguistics Society (CLS 1977)*, pages 651–668, Chicago, IL, 1977. Chicago Linguistics Society. [cited at p. 3]

[WM06] Yuk Wah Wong and Raymond J. Mooney. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL 2006)*, pages 439–446, 2006. [cited at p. 7, 10, 12, 95]

[WM07a] Yuk Wah Wong and Raymond J. Mooney. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL/HLT 2007)*, pages 172–179, 2007. [cited at p. 20, 21, 89, 99, 100, 102, 103, 113]

[WM07b] Yuk Wah Wong and Raymond J. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pages 960–967, 2007. [cited at p. 10]

[Won07] Yuk Wah Wong. *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques*. PhD thesis, The University of Texas at Austin, 2007. [cited at p. 2, 15, 23, 96, 97, 98, 102]

[YK01] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, pages 523–530, 2001. [cited at p. 41]

[ZC05] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005. [cited at p. 10, 18, 19, 108]

[ZC07] Luke S. Zettlemoyer and Michael Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 678–687, 2007. [cited at p. 10, 18, 19, 108]

[ZM96] John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI 1996)*, pages 1050–1055. AAAI Press, 1996. [cited at p. 3]