A Generative Model for Parsing Natural Language to Meaning Representations

Wei Lu[1], **Hwee Tou Ng**[1,2], **Wee Sun Lee**[1,2]
[1]Singapore-MIT Alliance
[2]Department of Computer Science
National University of Singapore
luwei@nus.edu.sg
{nght,leews}@comp.nus.edu.sg

Luke S. Zettlemoyer
CSAIL
Massachusetts Institute of Technology
lsz@csail.mit.edu

Abstract

In this paper, we present an algorithm for learning a generative model of natural language sentences together with their formal meaning representations with hierarchical structures. The model is applied to the task of mapping sentences to hierarchical representations of their underlying meaning. We introduce dynamic programming techniques for efficient training and decoding. In experiments, we demonstrate that the model, when coupled with a discriminative reranking technique, achieves state-of-the-art performance when tested on two publicly available corpora. The generative model degrades robustly when presented with instances that are different from those seen in training. This allows a notable improvement in recall compared to previous models.

1 Introduction

To enable computers to understand natural human language is one of the classic goals of research in natural language processing. Recently, researchers have developed techniques for learning to map sentences to hierarchical representations of their underlying meaning (Wong and Mooney, 2006; Kate and Mooney, 2006).

One common approach 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.

In this paper, we develop an alternative approach that learns a model which 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 *hybrid trees* whose nodes include both natural language words and meaning representation tokens. Inspired by the work of Collins (2003), 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 observe that it can correctly parse a wider range of test examples than previous approaches.

The generative model is learned from data that consists of sentences paired with their meaning representations. However, there is no explicit labeling of the correspondence between words and meaning tokens 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 inference, where n is the sentence length and m is the size of meaning structure. This approach allows for efficient training and decoding.

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.

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 783–792,
Honolulu, October 2008.

To address this problem, we use a simple reranking approach to select a parse from a k-best list of parses. This pipelined approach achieves state-of-the-art performance on two publicly available corpora. In particular, the flexible generative model leads to notable improvements in recall, the total percentage of sentences that are correctly parsed.

2 Related Work

In Section 9, we will compare performance with the three existing systems that were evaluated on the same data sets we consider. SILT (Kate et al., 2005) learns deterministic rules to transform either sentences or their syntactic parse trees to meaning structures. WASP (Wong and Mooney, 2006) is a system motivated by statistical machine translation techniques. It acquires a set of synchronous lexical entries by running the IBM alignment model (Brown et al., 1993) and learns a log-linear model to weight parses. KRISP (Kate and Mooney, 2006) is a discriminative approach where meaning representation structures are constructed from the natural language strings hierarchically. It is built on top of SVM^{struct} with string kernels.

Additionally, there is substantial related research that is not directly comparable to our approach. Some of this work requires different levels of supervision, including labeled syntactic parse trees (Ge and Mooney, 2005; Ge and Mooney, 2006). Others do not perform lexical learning (Tang and Mooney, 2001). Finally, recent work has explored learning to map sentences to lambda-calculus meaning representations (Wong and Mooney, 2007; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007).

3 Meaning Representation

We restrict our meaning representation (MR) formalism to a variable free version as presented in (Wong and Mooney, 2006; Kate et al., 2005).

A training instance consists of a natural language sentence (NL sentence) and its corresponding meaning representation structure (MR structure). Consider the following instance taken from the GEOQUERY corpus (Kate et al., 2005):

The NL sentence "`How many states do not have rivers ?`" consists of 8 words, including punctuation. The MR is a hierarchical tree structure, as shown in Figure 1.

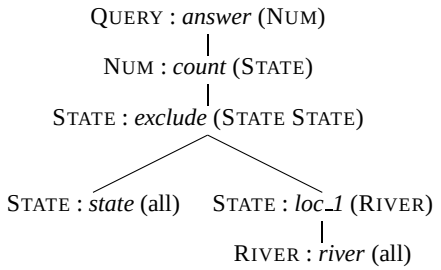

Figure 1: An example MR structure

Following an inorder 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) NUM : *count* (STATE)
(2) STATE : *exclude* (STATE_1 STATE_2)
(3) STATE : *state* (all)
(4) STATE : *loc_1* (RIVER)
(5) RIVER : *river* (all)

Each such MR production consists of three components: a *semantic category*, a *function symbol* which can be omitted (considered empty), and a list of *arguments*. 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" as its only argument. Production (5) has "RIVER" as its semantic category, "*river*" as the function symbol, and "all" is a constant.

4 The Generative Model

We describe in this section our proposed generative model, which simultaneously generates a NL sentence and an MR structure.

We denote a single NL word as w, a contiguous sequence of NL words as $\mathbf{w}$, and a complete NL sentence as $\widehat{\mathbf{w}}$. In the MR structure, we denote a semantic category as $\mathcal{M}$. We denote a single MR production as m_a, or $\mathcal{M}_a : p_\alpha(\mathcal{M}_b, \mathcal{M}_c)$, where $\mathcal{M}_a$ is the semantic category for this production, p_α is the function symbol, and $\mathcal{M}_b, \mathcal{M}_c$ are the child semantic categories. We denote $\mathbf{m}_a$ as an MR structure rooted by an MR production m_a, and $\widehat{\mathbf{m}_a}$ an MR structure for a complete sentence rooted by an MR production m_a.

The model generates a *hybrid tree* that represents a sentence $\widehat{\mathbf{w}} = \mathbf{w}_1 \ldots \mathbf{w}_2 \ldots$ paired with an MR structure $\widehat{\mathbf{m}_a}$ rooted by m_a.

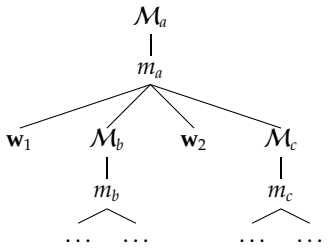

Figure 2: The generation process

Figure 2 shows part of a hybrid tree that is generated as follows. Given a semantic category $\mathcal{M}_a$, we first pick an MR production m_a that has the form $\mathcal{M}_a : p_\alpha(\mathcal{M}_b, \mathcal{M}_c)$, which gives us the function symbol p_α as well as the child semantic categories $\mathcal{M}_b$ and $\mathcal{M}_c$. Next, we generate the *hybrid sequence* of child nodes $\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}$, which consists of NL words and semantic categories.

After that, 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 a NL sentence $\mathbf{w}$ by recording the NL words visited in depth-first traversal order and can recover an MR structure $\mathbf{m}$ by following a tree-specific traversal order, defined by the hybrid-patterns we introduce below. Figure 3 gives a partial hybrid tree for the training example from Section 3. Note that the leaves of a hybrid tree are always NL tokens.

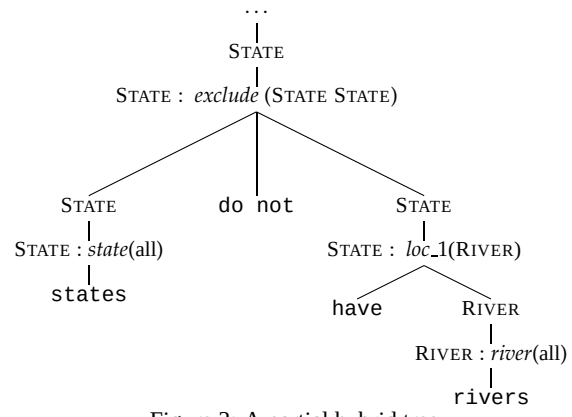

Figure 3: A partial hybrid tree

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

$$P(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T}) = P(\mathcal{M}_a) \times P(m_a|\mathcal{M}_a) \times P(\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}|m_a) \times P(m_b|m_a, \arg = 1) \times P(\ldots|m_b) \times P(m_c|m_a, \arg = 2) \times P(\ldots|m_c) \quad (1)$$

where "arg" refers to the position of the child semantic category in the argument list.

Motivated by Collins' syntactic parsing models (Collins, 2003), we consider the generation process for a hybrid sequence from an MR production as a Markov process.

Given the assumption that each MR production has at most two semantic categories in its arguments (any production can be transformed into a sequence of productions of this form), Table 1 includes the list of all possible *hybrid patterns*.

# RHS	Hybrid Pattern	# Patterns
0	$m \rightarrow \mathbf{w}$	1
1	$m \rightarrow [\mathbf{w}]\mathcal{Y}[\mathbf{w}]$	4
2	$m \rightarrow [\mathbf{w}]\mathcal{Y}[\mathbf{w}]\mathcal{Z}[\mathbf{w}]$	8
	$m \rightarrow [\mathbf{w}]\mathcal{Z}[\mathbf{w}]\mathcal{Y}[\mathbf{w}]$	8

Table 1: A list of hybrid patterns, [] denotes optional

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, and anything inside [] can be optionally omitted. The last row contains 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 STATE : *exclude* (STATE$_1$ STATE$_2$) generates a hybrid sequence $\overline{\text{STATE}_1\ \texttt{do not}\ \text{STATE}_2}$, the hybrid pattern $m \rightarrow \mathcal{Y}\mathbf{w}\mathcal{Z}$ is associated with this generation step.

For the example hybrid tree in Figure 2, we can decompose the probability for generating the hybrid sequence as follows:

$$P(\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}|m_a) = P(m \rightarrow \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}|m_a) \times P(\mathbf{w}_1|m_a) \times P(\mathcal{M}_b|m_a, \mathbf{w}_1) \times P(\mathbf{w}_2|m_a, \mathbf{w}_1, \mathcal{M}_b) \times P(\mathcal{M}_c|m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2) \times P(\texttt{END}|m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2, \mathcal{M}_c) \quad (2)$$

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 information that we condition on, the *context*.

Note that our generative model is different from the synchronous context free grammars (SCFG) in a number of ways. A standard SCFG produces a correspondence between a pair of trees while our model produces a single hybrid tree that represents

the correspondence between a sentence and a tree. Also, SCFGs use a finite set of context-free rewrite rules to define the model, where the rules are possibly weighted. In contrast, we make use of the more flexible Markov models at each level of the generative process, which allows us to potentially produce a far wider range of possible trees.

5 Parameter Estimation

There are three categories of parameters used in the model. The first category of parameters models the generation of new MR productions from their parent MR productions: *e.g.*, $P(m_b|m_a, \arg = 1)$; the second models the generation of a hybrid sequence from an MR production: *e.g.*, $P(\mathbf{w}_1|m_a)$, $P(\mathcal{M}_b|m_a, \mathbf{w}_1)$; the last models the selection of a hybrid pattern given an MR production, *e.g.*, $P(m \rightarrow \mathbf{w}\mathcal{Y}|m_a)$. We will estimate parameters from all categories, with the following constraints:

1. $\sum_{m'} \rho(m'|m_j, \arg{=}k) = 1$ for all j and $k = 1, 2$.

   These parameters model the MR structures, and can be referred to as *MR model parameters*.

2. $\sum_t \theta(t|m_j, \Lambda) = 1$ for all j, where t is a NL word, the "END" symbol, or a semantic category. Λ is the context associated with m_j and t.

   These parameters model the emission of NL words, the "END" symbol, and child semantic categories from an MR production. We call them *emission parameters*.

3. $\sum_r \phi(r|m_j) = 1$ for all j, where r is a hybrid pattern listed in Table 1.

   These parameters model the selection of hybrid patterns. We name them *pattern parameters*.

With different context assumptions, we reach different variations of the model. In particular, we consider three assumptions, as follows:

Model I We make the following assumption:

$$\theta(t_k|m_j, \Lambda) = P(t_k|m_j) \quad (3)$$

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

In other words, generation of the next NL word depends on its direct parent MR production only. Such a *Unigram Model* may help in recall (the number of correct outputs over the total number of inputs), because it requires the least data to estimate.

Model II We make the following assumption:

$$\theta(t_k|m_j, \Lambda) = P(t_k|m_j, t_{k-1}) \quad (4)$$

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, 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. This model is also referred to as *Bigram Model*. This model may help in precision (the number of correct outputs over the total number of outputs), because it conditions on a larger context.

Model III We make the following assumption:

$$\theta(t_k|m_j, \Lambda) = \frac{1}{2} \times \Big(P(t_k|m_j) + P(t_k|m_j, t_{k-1})\Big) \quad (5)$$

We can view this model, called the *Mixgram Model*, as an interpolation between Model I and II. This model gives us a balanced score for both precision and recall.

5.1 Modeling Meaning Representation

The MR model parameters can be estimated independently from the other two. These parameters can be viewed as the "language model" parameters for the MR structure, and can be estimated directly from the corpus by simply reading off the counts of occurrences of MR productions in MR structures over the training corpus. To resolve data sparseness problem, a variant of the bigram Katz Back-Off Model (Katz, 1987) is employed here for smoothing.

5.2 Learning the Generative Parameters

Learning the remaining two categories of parameters is more challenging. In a conventional PCFG parsing task, 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 structures is unknown. Many possible derivations could reach the same NL-MR pair, where each such derivation forms a hybrid tree.

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 (Yamada and Knight, 2001), as discussed next.

5.2.1 The Inside-Outside Algorithm with EM

In this section, we discuss how to estimate the emission and pattern parameters with the Expectation Maximization (EM) algorithm (Dempster et al., 1977), by using an inside-outside (Baker, 1979) dynamic programming approach.

Denote $\mathbf{n}^i \equiv \langle \mathbf{m}^i, \mathbf{w}^i \rangle$ as the i-th training instance, where $\mathbf{m}^i$ and $\mathbf{w}^i$ are the MR structure and the NL sentence of the i-th instance respectively. We also denote $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ as an aligned pair of MR substructure and contiguous NL substring, where the MR substructure rooted by MR production m_v will correspond to (*i.e.*, hierarchically generate) the NL substring $\mathbf{w}_v$. The symbol h is used to denote a hybrid sequence, and the function *Parent*(h) gives the unique MR substructure-NL subsequence pair which can be decomposed as h. *Parent*($\mathbf{n}_v$) returns the set of all possible hybrid sequences under which the pair $\mathbf{n}_v$ can be generated. Similarly, *Children*(h) gives the NL-MR pairs that appear directly below the hybrid sequence h in a hybrid tree, and *Children*($\mathbf{n}$) returns the set of all possible hybrid sequences that $\mathbf{n}$ can be decomposed as. Figure 4 gives a packed tree structure representing the relations between the entities.

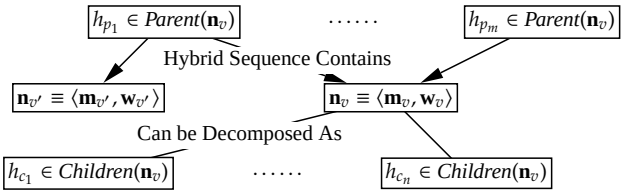

Figure 4: A packed tree structure representing the relations between hybrid sequences and NL-MR pairs

The formulas for computing inside and outside probabilities as well as the equations for updating parameters are given in Figure 5. We use a CKY-style parse chart for tracking the probabilities.

5.2.2 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 is deployed. We assume half of the time words can

The inside (β) probabilities are defined as

- If $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ is leaf

$$\beta(\mathbf{n}_v) = P(\mathbf{w}_v | \mathbf{m}_v) \tag{6}$$

- If $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ is not leaf

$$\beta(\mathbf{n}_v) = \sum_{h \in Children(\mathbf{n}_v)} \Big(P(h|\mathbf{m}_v) \times \prod_{\mathbf{n}_{v'} \in Children(h)} \beta(\mathbf{n}_{v'}) \Big) \tag{7}$$

The outside (α) probabilities are defined as

- If $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ is root

$$\alpha(\mathbf{n}_v) = 1 \tag{8}$$

- If $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ is not root

$$\alpha(\mathbf{n}_v) = \sum_{h \in Parent(\mathbf{n}_v)} \Big(\alpha\big(Parent(h)\big) \times P\big(h|Parent(h)\big) \times \prod_{\mathbf{n}_{v'} \in Children(h), v' \neq v} \beta(\mathbf{n}_{v'}) \Big) \tag{9}$$

Parameter Update

- Update the emission parameter

  The count $c^i(t, m_v, \Lambda_k)$, where t is a NL word or a semantic category, for an instance pair $\mathbf{n}^i \equiv \langle \mathbf{m}^i, \mathbf{w}^i \rangle$:

$$c^i(t, m_v, \Lambda_k) = \frac{1}{\beta(\mathbf{n}^i)} \times \sum_{(t, m_v, \Lambda_k) \text{ in } h \in Children(m_v)} \Big(\alpha(\mathbf{n}^i_v) \times P(h|m_v) \times \prod_{\mathbf{n}^i_{v'} \in Children(h)} \beta(\mathbf{n}^i_{v'}) \Big)$$

  The emission parameter is re-estimated as:

$$\theta'(t|m_v, \Lambda_k) = \frac{\sum_i c^i(t, m_v, \Lambda_k)}{\sum_{t'} \sum_i c^i(t', m_v, \Lambda_k)} \tag{10}$$

- Update the pattern parameter

  The count $c^i(r, m_v)$, where r is a hybrid pattern, for an instance pair $\mathbf{n}^i \equiv \langle \mathbf{m}^i, \mathbf{w}^i \rangle$:

$$c^i(r, m_v) = \frac{1}{\beta(\mathbf{n}^i)} \times \sum_{(r, m_v) \text{ in } h \in Children(m_v)} \Big(\alpha(\mathbf{n}^i_v) \times P(h|m_v) \times \prod_{\mathbf{n}^i_{v'} \in Children(h)} \beta(\mathbf{n}^i_{v'}) \Big)$$

  The pattern parameter is re-estimated as:

$$\phi'(r|m_v) = \frac{\sum_i c^i(r, m_v)}{\sum_{r'} \sum_i c^i(r', m_v)} \tag{11}$$

Figure 5: The inside/outside formulas as well as update equations for EM

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:

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

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

6 A Dynamic Programming Algorithm for Inside-Outside Computation

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 and m are the length of the NL sentence and the size of the MR structure respectively. This is not very practical as in one of the corpora we look at, n and m can be up to 45 and 20 respectively.

In this section, we develop an efficient dynamic programming algorithm that enables the inference to run in $O(n^3m)$ time. The idea is as follows. Instead of treating each possible hybrid sequence as a separate rule, we efficiently aggregate the already computed probability scores for hybrid sequences that share identical hybrid patterns. Such aggregated scores can then be used for subsequent computations. By doing this, we can effectively avoid a large amount of redundant computations. The algorithm supports both unigram and bigram context assumptions. For clarity and ease of presentation, we primarily make the unigram assumption throughout our discussion.

We use $\beta(\mathbf{m}_v, \mathbf{w}_v)$ to denote the inside probability for $\mathbf{m}_v$-$\mathbf{w}_v$ pair, $b_r[\mathbf{m}_v, \mathbf{w}_v, c]$ to denote the aggregated probabilities for the MR sub-structure $\mathbf{m}_v$ to generate all possible hybrid sequences based on $\mathbf{w}_v$ with pattern r that covers its c-th child only. In addition, we use $\mathbf{w}_{(i,j)}$ to denote a subsequence of $\mathbf{w}$ with start index i (inclusive) and end index j (exclusive). We also use $\beta_r[\![\mathbf{m}_v, \mathbf{w}_v]\!]$ to denote the aggregated inside probability for the pair $\langle \mathbf{m}_v, \mathbf{w}_v \rangle$, if the hybrid pattern is restricted to r only. By definition we have:

$$\beta(\mathbf{m}_v, \mathbf{w}_v) = \sum_r \phi(r|m_v) \times \beta_r[\![\mathbf{m}_v, \mathbf{w}_v]\!] \times \theta(\texttt{END}|m_v) \quad (12)$$

Relations between β_r and b_r can also be established. For example, if m_v has one child semantic category, we have:

$$\beta_{m \to \mathbf{w}\mathcal{Y}}[\![\mathbf{m}_v, \mathbf{w}_v]\!] = b_{m \to \mathbf{w}\mathcal{Y}}[\mathbf{m}_v, \mathbf{w}_v, 1] \quad (13)$$

For the case when m_v has two child semantic categories as arguments, we have, for example:

$$\beta_{m \to \mathbf{w}\mathcal{Y}\mathcal{Z}\mathbf{w}}[\![\mathbf{m}_v, \mathbf{w}_{(i,j)}]\!] = \sum_{i+2 \le k \le j-2} b_{m \to \mathbf{w}\mathcal{Y}}[m_v, \mathbf{w}_{(i,k)}, 1] \times b_{m \to \mathcal{Y}\mathbf{w}}[m_v, \mathbf{w}_{(k,j)}, 2] \quad (14)$$

Note that there also exist relations amongst b terms for more efficient computation, for example:

$$b_{m \to \mathbf{w}\mathcal{Y}}[\mathbf{m}_v, \mathbf{w}_{(i,j)}, c] = \theta(w_i|m_v) \times \Big(b_{m \to \mathbf{w}\mathcal{Y}}[\mathbf{m}_v, \mathbf{w}_{(i+1,j)}, c] + b_{m \to \mathcal{Y}}[\mathbf{m}_v, \mathbf{w}_{(i+1,j)}, c] \Big) \quad (15)$$

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.

7 Decoding

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

$$\widehat{\mathbf{m}}^* = \arg\max_{\widehat{\mathbf{m}}} P(\widehat{\mathbf{m}}|\widehat{\mathbf{w}}) = \arg\max_{\widehat{\mathbf{m}}} \sum_{\mathcal{T}} P(\widehat{\mathbf{m}}, \mathcal{T}|\widehat{\mathbf{w}}) \quad (16)$$

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

$$\widehat{\mathbf{m}}^* = \arg\max_{\widehat{\mathbf{m}}} \max_{\mathcal{T}} P(\widehat{\mathbf{m}}, \mathcal{T}|\widehat{\mathbf{w}}) = \arg\max_{\widehat{\mathbf{m}}} \max_{\mathcal{T}} P(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T}) \quad (17)$$

We have implemented an exact top-k decoding algorithm for this task. Dynamic programming techniques similar to those discussed in Section 6 can also be applied when retrieving the top candidates.

We also find the Viterbi hybrid tree given a NL-MR pair, which can be done in an analogous way. This tree will be useful for reranking.

8 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.

Feature Type	Description	Example
1. Hybrid Rule	A MR production and its child hybrid form	f_1 : STATE : *loc_1*(RIVER) → `have` RIVER
2. Expanded Hybrid Rule	A MR production and its child hybrid form expanded	f_2 : STATE : *loc_1*(RIVER) → ⟨`have`, RIVER : *river*(all)⟩
3. Long-range Unigram	A MR production and a NL word appearing below in tree	f_3 : STATE : *exclude*(STATE STATE) → `rivers`
4. Grandchild Unigram	A MR production and its grandchild NL word	f_4 : STATE : *loc_1*(RIVER) → `rivers`
5. Two Level Unigram	A MR production, its parent production, and its child NL word	f_5 : ⟨RIVER : *river*(all), STATE : *loc_1*(RIVER)⟩ → `rivers`
6. Model Log-Probability	Logarithm of base model's joint probability	$\log\left(P(\mathbf{w}, \mathbf{m}, \mathcal{T})\right)$.

Table 2: All the features used. There is one feature for each possible combination, under feature type 1-5. It takes value 1 if the combination is present, and 0 otherwise. Feature 6 takes real values.

8.1 The Averaged Perceptron Algorithm with Separating Plane

The averaged perceptron algorithm (Collins, 2002) has previously been applied to various NLP tasks (Collins, 2002; Collins, 2001) for discriminative reranking. The detailed algorithm can be found in (Collins, 2002). 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 determined by running the Viterbi algorithm on each training NL-MR pair. The feature function is discussed in section 8.2.

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.

8.2 Features

We list in Table 2 the set of features we used. Examples are given based on the hybrid tree in Figure 3. Some of the them are adapted from (Collins and Koo, 2005) 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 in Table 2 is present, 0 otherwise), while feature 6 is real valued. Features that do not appear more than once in the training set are discarded.

9 Evaluation

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 (2006).

9.1 Training Methodology

Given a training set, we first run a variant of IBM alignment model 1 (Brown et al., 1993) for 100 iterations, and then initialize Model I 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 Model I for 100 EM iterations and use the learned parameters to initialize Model II which is trained for another 100 EM iterations. Model III 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.

9.2 Evaluation Methodology

Following Wong (2007) and other previous work, we report performance in terms of *Precision* (percentage of answered NL sentences that are correct), *Recall* (percentage of correctly answered NL sentences, out of all NL sentences) and *F*-score (harmonic mean of *Precision* and *Recall*).

Again following Wong (2007), 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.

9.3 Comparison over Three Models

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

Table 3: Performance comparison over three models (*Prec.*:precision, *Rec.*:recall, +R: with reranking)

We evaluated the three models, with and without reranking. The results are presented in Table 3. Comparing Model I and Model II, we noticed that for both corpora, Model I in general achieves better recall while Model II achieves better precision. This observation conforms to our earlier expectations. Model III, as an interpolation of the above two models, achieves a much better *F*-measure on GEOQUERY corpus. However, it is shown to be less effective on ROBOCUP corpus. We noticed that compared to the GEOQUERY corpus, 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 Model III 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 Model I or Model II. 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.

9.4 Comparison with Other 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 4, where the performance scores for the previous systems are taken from (Wong, 2007). For 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[1].

System	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
Model III+R	89.3	**81.5**	**85.2**	82.5	**67.7**	**74.4**

Table 4: Performance comparison with other directly comparable systems

9.5 Performance on Other 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

[1]We are unable to perform statistical significance tests because the detailed performance for each fold of previously published research work is not available.

the GEOQUERY corpus consisting of 250 instances, with four different NL annotations.

As we can see from Table 5, our model is able to achieve performance comparable to WASP as reported by Wong (2007).

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

Table 5: Performance on different natural languages for GEOQUERY-250 corpus

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

10 Conclusions

We presented a new generative model that simultaneously produces both NL sentences and their corresponding MR structures. The model can be effectively applied to the task of transforming NL sentences to their MR structures. We also developed a new dynamic programming algorithm for efficient training and decoding. We demonstrated that this approach, augmented with a discriminative reranking technique, achieves state-of-the-art performance when tested on standard benchmark corpora.

In future, we would like to extend the current model to have a wider range of support of MR formalisms, such as the one with lambda-calculus support. We are also interested in investigating ways to apply the generative model to the inverse task: generation of a NL sentence that explains a given MR structure.

Acknowledgments

The authors would like to thank Leslie Pack Kaelbling for her valuable feedback and comments on this research. The authors would also like to thank the anonymous reviewers for their thoughtful comments on this paper. The research is partially supported by ARF grant R-252-000-240-112.

References

J. K. Baker. 1979. Trainable grammars for speech recognition. *Journal of the Acoustical Society of America*, 65:S132.

P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

M. Collins and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

M. Collins. 2001. 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.

M. Collins. 2002. 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.

M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

R. Ge and R. J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL 2005)*, pages 9–16.

R. Ge and R. J. Mooney. 2006. 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.

R. J. Kate and R. J. Mooney. 2006. 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.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, pages 1062–1068.

S. Katz. 1987. 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.

L. R. Tang and R. J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning (ECML 2001)*, pages 466–477.

Y. W. Wong and R. J. Mooney. 2006. 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.

Y. W. Wong and R. J. Mooney. 2007. 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.

Y. W. Wong. 2007. *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques*. Ph.D. thesis, The University of Texas at Austin.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*.

L. S. Zettlemoyer and M. Collins. 2007. 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.