Semantic Parsing with Relaxed Hybrid Trees

Wei Lu
Information Systems Technology and Design
Singapore University of Technology and Design
luwei@sutd.edu.sg

Abstract

We propose a novel model for parsing natural language sentences into their formal semantic representations. The model is able to perform integrated lexicon acquisition and semantic parsing, mapping each atomic element in a complete semantic representation to a contiguous word sequence in the input sentence in a recursive manner, where certain overlappings amongst such word sequences are allowed. It defines distributions over the novel *relaxed hybrid tree* structures which jointly represent both sentences and semantics. Such structures allow tractable dynamic programming algorithms to be developed for efficient learning and decoding. Trained under a discriminative setting, our model is able to incorporate a rich set of features where certain unbounded long-distance dependencies can be captured in a principled manner. We demonstrate through experiments that by exploiting a large collection of simple features, our model is shown to be competitive to previous works and achieves state-of-the-art performance on standard benchmark data across four different languages. The system and code can be downloaded from http://statnlp.org/research/sp/.

1 Introduction

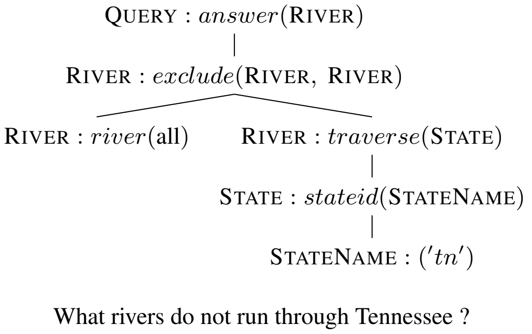
QUERY : $answer$(RIVER)
RIVER : $exclude$(RIVER, RIVER)
RIVER : $river$(all)
RIVER : $traverse$(STATE)
STATE : $stateid$(STATENAME)
STATENAME : ($'tn'$)
What rivers do not run through Tennessee ?

Figure 1: An example tree-structured semantic representation (above) and its corresponding natural language sentence.

Semantic parsing, the task of transforming natural language sentences into formal representations of their underlying semantics, is one of the classic goals for natural language processing and artificial intelligence. This area of research recently has received a significant amount of attention. Various models have been proposed over the past few years (Zettlemoyer and Collins, 2005; Kate and Mooney, 2006; Wong and Mooney, 2006; Lu et al., 2008; Jones et al., 2012).

Following previous research efforts, we perform semantic parsing under a setting where the semantics for complete sentences are provided as training data, but detailed word-level semantic information is not explicitly given during the training phase. As one example, consider the following natural language sentence paired with its corresponding semantic representation:

What rivers do not run through Tennessee ?
$answer(exclude(river(all), traverse(stateid('tn'))))$

The training data consists of a set of sentences paired with semantic representations. Our goal is to learn from such pairs a model, which can be effectively used for parsing novel sentences into their semantic representations.

Certain assumptions about the semantics are typically made. One common assumption is that the semantics can be represented as certain recursive structures such as trees, which consist of atomic semantic units as tree nodes. For example, the above semantics can be converted into an equivalent tree structure as illustrated in Figure 1. We will provide more details about such tree structured semantic representations in Section 2.1.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1308–1318,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

Currently, most state-of-the-art approaches that deal with such tree structured semantic representations either cast the semantic parsing problem as a statistical string-to-string transformation problem (Wong and Mooney, 2006), which ignores the potentially useful structural information of the tree, or employ latent-variable models to capture the correspondences between words and tree nodes using a generative approach (Lu et al., 2008; Jones et al., 2012). While generative models can be used to flexibly model the correspondences between individual words and semantic nodes of the tree, such an approach is limited to modeling local dependencies and is unable to flexibly incorporate a large set of potentially useful features.

In this work, we propose a novel model for parsing natural language into tree structured semantic representations. Specifically, we propose a novel *relaxed hybrid tree* representation which jointly encodes both natural language sentences and semantics; such representations can be effectively learned with a latent-variable discriminative model where long-distance dependencies can be captured. We present dynamic programming algorithms for efficient learning and decoding. With a large collection of simple features, our model reports state-of-the-art results on benchmark data annotated with four different languages.

Furthermore, although we focus our discussions on semantic parsing in this work, our proposed model is a general. Essentially our model is a discriminative string-to-tree model which recursively maps overlapping contiguous word sequences to tree nodes at different levels, where efficient dynamic programming algorithms can be used. Such a model may find applications in other areas of natural language processing, such as statistical machine translation and information extraction.

2 Background

2.1 Semantics

Various semantic formalisms have been considered for semantic parsing. Examples include the tree-structured semantic representations (Wong and Mooney, 2006), the lambda calculus expressions (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007), and dependency-based compositional semantic representations (Liang et al., 2013). In this work, we specifically focus on the tree-structured representations for semantics.

Each semantic representation consists of semantic units as its tree nodes, where each semantic unit is of the following form:

$$m_a \equiv \tau_a : p_\alpha(\tau_b *) \qquad (1)$$

Here m_a is used to denote a complete semantic unit, which consists of its semantic type τ_a, its function symbol p_α, as well as an argument list $\tau_b *$ (we assume there are at most two arguments for each semantic unit). In other words, each semantic unit can be regarded as a function which takes in other semantics of specific types as arguments, and returns new semantics of a particular type. For example, in Figure 1, the semantic unit at the root has a type QUERY, a function name $answer$, and a single argument type RIVER.

2.2 Joint Representations

Semantic parsing models transform sentences into their corresponding semantics. It is therefore essential to make proper assumptions about joint representations for language and semantics that capture how individual words and atomic semantic units connect to each other. Typically, different existing models employ different assumptions for establishing such connections, leading to very different definitions of joint representations. We survey in this section various representations proposed by previous works.

The WASP semantic parser (Wong and Mooney, 2006) essentially casts the semantic parsing problem as a string-to-string transformation problem by employing a statistical phrase-based machine translation approach with synchronous grammars (Chiang, 2007). Therefore, one can think of the joint representation for both language and semantics as a synchronous derivation tree consisting of those derivation steps for transforming sentences into target semantic representation strings. While this joint representation is flexible, allowing blocks of semantic structures to map to word sequences, it does not fully exploit the structural information (tree) as conveyed by the semantics.

The KRISP semantic parser (Kate and Mooney, 2006) makes use of Support Vector Machines with string kernels (Lodhi et al., 2002) to recursively map contiguous word sequences into semantic units to construct a tree structure. Our *relaxed hybrid tree* structures also allow input word sequences to map to semantic units in a recursive manner. One key distinction, as we will see, is that our structure distinguishes words which are immediately associated with a particular semantic unit, from words which are remotely associated.

The SCISSOR model (Ge and Mooney, 2005) performs integrated semantic and syntactic parsing. The model parses natural language sentences into semantically augmented parse trees whose nodes consist of both semantic and syntactic labels and then builds semantic representations based on such augmented trees. Such a joint representation conveys more information, but requires language-specific syntactic analysis.

The *hybrid tree* model (Lu et al., 2008) is based on the assumption that there exists an underlying generative process which jointly produces both the sentence and the semantic tree in a top-down recursive manner. The generative process results in a hybrid tree structure which consists of words as leaves and semantic units as nodes. An example hybrid tree structure is shown in Figure 2 (a). Such a representation allows each semantic unit to map to a possibly *discontiguous* sequence of words. The model was shown to be effective empirically, but it implicitly assumes that both the sentence and semantics exhibit certain degree of structural similarity that allows the hybrid tree structures to be constructed.

UBL (Kwiatkowski et al., 2010) is a semantic parser based on restricted higher-order unification with CCG (Steedman, 1996). The model can be used to handle both tree structured semantic representations and lambda calculus expressions, and assumes there exist CCG derivations as joint representations in which each semantic unit is associated with a contiguous word sequence where overlappings amongst word sequences are not allowed.

Jones et al. (2012) recently proposed a framework that performs semantic parsing with tree transducers. The model learns representations that are similar to the hybrid tree structures using a generative process under a Bayesian setting. Thus, their representations also potentially present similar issues as the ones mentioned above.

Besides these supervised approaches, recently there are also several works that take alternative learning approaches to (mostly task-dependent) semantic parsing. Poon and Domingos (2009) proposed a model for unsupervised semantic parsing that transforms dependency trees into semantic representations using Markov logic (Richardson and Domingos, 2006). Clarke et al. (2010) proposed a model that learns a semantic parser for answering questions without relying on semantic annotations. Goldwasser et al. (2011) took an unsupervised approach for semantic parsing based on self-training driven by confidence estimation. Liang et al. (2013) proposed a model for learning the dependency-based compositional semantics (DCS) which can be used for optimizing the end-task performance. Artzi and Zettlemoyer (2013) proposed a model for mapping instructions to actions with weak supervision.

Symbol	Description
$\mathbf{n}$	A complete natural language sentence
$\mathbf{m}$	A complete semantic representation
$\mathbf{h}$	A complete latent joint representation (or, a *relaxed hybrid tree* for our work)
$\mathcal{H}(\mathbf{n},\mathbf{m})$	A complete set of latent joint representations that contain the $(\mathbf{n},\mathbf{m})$ pair exactly
n, n_a	A contiguous sequence of words
w, w_k	A natural language word
m, m_a	A semantic unit
h, h_a	A node in the relaxed hybrid tree
Φ	The feature vector
ϕ_k	The k-th feature
Λ	The weight vector (model parameters)
λ_k	The weight for the k-th feature ϕ_k

Table 1: Notation Table

3 Approach

We discuss our approach to semantic parsing in this section. The notation that we use in this paper is summarized in Table 1.

3.1 Model

In standard supervised syntactic parsing, one typically has access to a complete syntactic parse tree for each sentence in the training phase, which exactly tells the correct associations between words and syntactic labels. In our problem, however, each sentence is only paired with a complete semantic representation where the correct associations between words and semantic units are unavailable. We thus need to model such information with latent variables.

For a given $\mathbf{n}$-$\mathbf{m}$ pair (where $\mathbf{n}$ is a complete natural language sentence, and $\mathbf{m}$ is a complete semantic representation), we assume there exists a latent joint representation $\mathbf{h}$ that consists of both $\mathbf{n}$ and $\mathbf{m}$ which tells the correct associations between words and semantic units in such a pair. We use $\mathcal{H}(\mathbf{n},\mathbf{m})$[1] to denote the set of all such possible

[1] We will give a concrete definition of $\mathcal{H}(\mathbf{n},\mathbf{m})$ used for this work, which is the complete set of all possible *relaxed hybrid tree* structures for the $\mathbf{n}$-$\mathbf{m}$ pair, when we discuss our own joint representations later in Section 3.2.

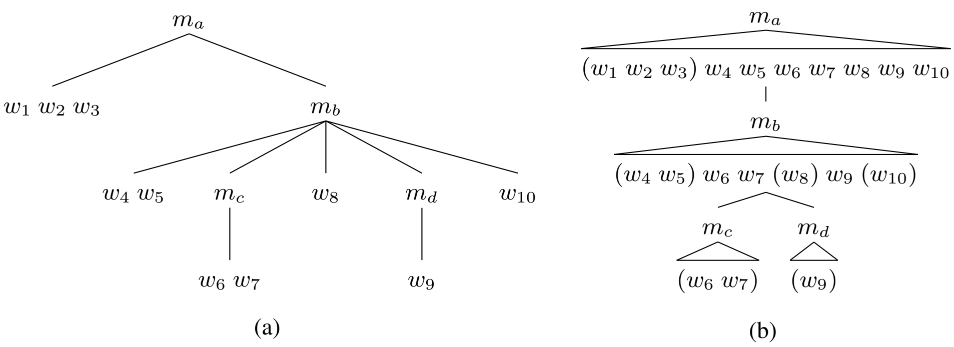

Figure 2: Two different ways of jointly representing sentences and their semantics. The hybrid tree representation of Lu et al. (2008) (left), and our novel *relaxed hybrid tree* representation (right). In our representation, a word w can be either *immediately* associated with its parent m (the words which appear inside the parenthesis), or *remotely* associated with m (the words that do not appear inside the parenthesis, and will also appear under a subtree rooted by one of m's children).

latent joint representations that contain both $\mathbf{n}$ and $\mathbf{m}$ exactly.

Given the joint representations, to model how the data is generated, one can either take a generative approach which models the joint probability distribution over $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ tuples, or a discriminative approach which models the distribution over $(\mathbf{m}, \mathbf{h})$ tuples given the observation $\mathbf{n}$. Following several previous research efforts (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010; Liang et al., 2013), in this work we define a discriminative model using a log-linear approach:

$$P(\mathbf{m}, \mathbf{h}|\mathbf{n}; \Lambda) = \frac{e^{\Lambda \cdot \Phi(\mathbf{n},\mathbf{m},\mathbf{h})}}{\sum_{\mathbf{m}',\mathbf{h}' \in \mathcal{H}(\mathbf{n},\mathbf{m}')} e^{\Lambda \cdot \Phi(\mathbf{n},\mathbf{m}',\mathbf{h}')}} \quad (2)$$

Here $\Phi(\mathbf{n}, \mathbf{m}, \mathbf{h})$ is a function defined over the tuple $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ that returns a vector consisting of counts of features associated with the tuple, and Λ is a vector consisting of feature weights, which are the parameters of the model.

In practice, we are only given the $\mathbf{n}$-$\mathbf{m}$ pairs but the latent structures are not observed. We therefore consider the following marginal probability:

$$\begin{aligned} P(\mathbf{m}|\mathbf{n}; \Lambda) &= \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n},\mathbf{m})} P(\mathbf{m}, \mathbf{h}|\mathbf{n}; \Lambda) \\ &= \frac{\sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n},\mathbf{m})} e^{\Lambda \cdot \Phi(\mathbf{n},\mathbf{m},\mathbf{h})}}{\sum_{\mathbf{m}',\mathbf{h}' \in \mathcal{H}(\mathbf{n},\mathbf{m}')} e^{\Lambda \cdot \Phi(\mathbf{n},\mathbf{m}',\mathbf{h}')}} \quad (3) \end{aligned}$$

The above probability is defined for a particular $\mathbf{n}$-$\mathbf{m}$ pair. The complete log-likelihood objective for the training set is:

$$\begin{aligned} \mathcal{L}(\Lambda) &= \sum_i \log P(\mathbf{m}_i|\mathbf{n}_i; \Lambda) - \kappa||\Lambda||^2 \\ &= \sum_i \log \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}_i,\mathbf{m}_i)} P(\mathbf{m}_i, \mathbf{h}|\mathbf{n}_i; \Lambda) - \kappa||\Lambda||^2 \quad (4) \end{aligned}$$

where $(\mathbf{n}_i, \mathbf{m}_i)$ refers to the i-th instance in the training set. Note that here we introduce the additional regularization term $-\kappa \cdot ||\Lambda||^2$ to control over-fitting, where κ is a positive scalar.

Our goal is to maximize this objective function by tuning the model parameters Λ. Let's assume $\Lambda = \langle \lambda_1, \lambda_2, \ldots, \lambda_N \rangle$, where N is the total number of features (or the total number of parameters). Differentiating with respect to λ_k, the weight associated with the k-th feature ϕ_k, yields:

$$\begin{aligned} \frac{\partial \mathcal{L}(\Lambda)}{\partial \lambda_k} &= \sum_i \sum_{\mathbf{h}} \mathbf{E}_{P(\mathbf{h}|\mathbf{n}_i,\mathbf{m}_i;\Lambda)}[\phi_k(\mathbf{n}_i, \mathbf{m}_i, \mathbf{h})] \\ &- \sum_i \sum_{\mathbf{m},\mathbf{h}} \mathbf{E}_{P(\mathbf{m},\mathbf{h}|\mathbf{n}_i;\Lambda)}[\phi_k(\mathbf{n}_i, \mathbf{m}, \mathbf{h})] - 2\kappa\lambda_k \quad (5) \end{aligned}$$

where $\phi_k(\mathbf{n}, \mathbf{m}, \mathbf{h})$ refers to the number of occurrences for the k-th feature in the tuple $(\mathbf{n}, \mathbf{m}, \mathbf{h})$.

Given the objective value (4) and gradients (5), standard methods such as stochastic gradient descent or L-BFGS (Liu and Nocedal, 1989) can be employed to optimize the objective function. We will discuss the computation of the objective function and gradients next.

3.2 Relaxed Hybrid Trees

To allow tractable computation of the values for the objective function (4) and the gradients (5),

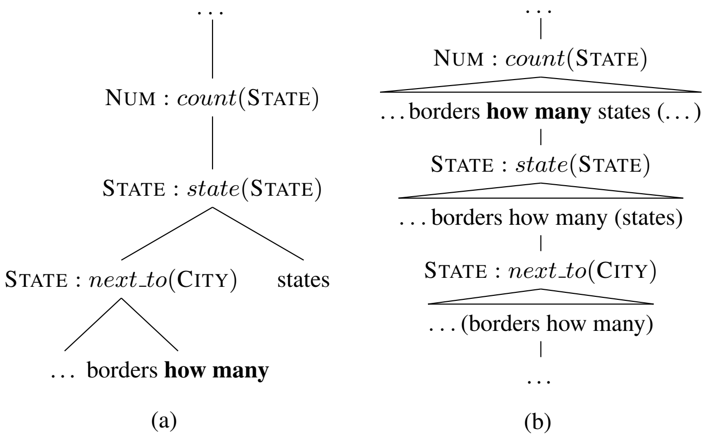

Figure 3: An example hybrid tree and an example *relaxed hybrid tree* representation. When the correct latent structure can not be found, the dependency between the words "how many" and the semantic unit "NUM : $count$(STATE)" can not be captured if the hybrid tree is used, whereas with our relaxed hybrid tree representation, such a dependency can still be captured.

certain restrictions on the latent structures ($\mathbf{h}$) will need to be imposed. We define in this section the set of all valid latent structures $\mathcal{H}(\mathbf{n}, \mathbf{m})$ for the $(\mathbf{n}, \mathbf{m})$ pair so that some efficient dynamic programming algorithms can be deployed.

We introduce our novel *relaxed hybrid tree* representations which jointly encode both natural language sentences and the tree-structured semantics. A *relaxed hybrid tree* $\mathbf{h}$ defined over $(\mathbf{n}, \mathbf{m})$ is a tree whose nodes are (n, m) pairs, where each n is a contiguous sequence of words from $\mathbf{n}$, and each m is a semantic unit (a tree node) from $\mathbf{m}$. For any two nodes $h_a \equiv (n_a, m_a)$ and $h_b \equiv (n_b, m_b)$ that appear in the relaxed hybrid tree $\mathbf{h}$, if h_a is the parent of h_b in $\mathbf{h}$, then m_a must also be the parent of m_b in $\mathbf{m}$, and n_a must contain n_b. If the lowest common ancestor of h_a and h_b in $\mathbf{h}$ is neither h_a nor h_b, then n_a and n_b do not share any common word. Note that words that appear at different positions in $\mathbf{n}$ are regarded as different words, regardless of their string forms.

Figure 2 (b) shows an example *relaxed hybrid tree* structure that we consider. Assume we would like to jointly represent both the natural language sentence $\mathbf{n} \equiv w_1 w_2 \dots\ w_{10}$ and its corresponding semantic representation $\mathbf{m} \equiv m_a(m_b(m_c, m_d))$. In the given example, the semantic unit m_a maps to the complete sentence, m_b maps to the sequence $w_4 w_5 \dots w_{10}$, m_c maps to $w_6 w_7$, and m_d maps to w_9. Certain words such as w_4 and w_{10} that appear directly below the semantic unit m_b but do not map to any of m_b's child semantic units are highlighted with parentheses "()", indicating they are *immediately associated* with m_b. These words play unique roles in the sub-tree rooted by m_b and are expected to be semantically closely related to m_b. Note that each word is immediately associated with *exactly one* semantic unit.

As a comparison, we also show an example hybrid tree representation (Lu et al., 2008) in Figure 2 (a) that has similar words-semantics correspondences. Different from our representation, the hybrid tree representation assumes each natural language word only maps to a single semantic unit (which is its immediate parent), and each semantic unit maps to a possibly discontiguous sequence of words. We believe that such a representation is overly restrictive, which might exhibit problems in cases where natural language sentences are highly non-isomorphic to their semantic tree structures. Under our relaxed hybrid tree representations, words that are immediately associated with a particular semantic unit now can also be *remotely associated* with all its parent semantic units as well. Essentially, our representation allows us to capture certain *unbounded* dependencies – for any word, as long as it appears below a certain semantic unit (in the relaxed hybrid tree), we can always capture the dependency between the two, regardless of which actual semantic unit that word is immediately associated with. Such an important relaxation allows some long-distance dependencies to be captured, which can potentially alleviate the sentence-semantics non-isomorphism issue reported in several earlier semantic parsing works (Kate and Mooney, 2006;

Wong and Mooney, 2007).

To better illustrate the differences, we show a concrete example in Figure 3, where the correct latent structure showing the correspondences between words and semantic units can not be found with the hybrid tree model. As a result, the hybrid tree model will fail to capture the correct dependency between the words "how many" and the semantic unit "$\textsc{Num} : count(\textsc{State})$". On the other hand, with our relaxed hybrid tree representation, such a dependency can still be captured, since these words will still be (remotely) associated with the semantic unit.

Such a relaxed hybrid tree representation, when further constrained with the word association patterns that we will introduce next, allows both the objective function (4) and the gradients of (5) to be computed through the dynamic programming algorithms to be presented in Section 4.

3.3 Word Association Patterns

As we have mentioned above, in the relaxed hybrid tree structures, each word w under a certain semantic unit m can either appear directly below m only (immediately associated with m), or can also appear in a subtree rooted by one of m's child semantic unit (remotely associated with m).

We allow several different ways for word associations and define the allowable patterns for semantic units with different number of arguments in Table 2. Such patterns are defined so that our model is amendable to dynamic programming algorithms to be discussed in Sec 4. In this table, $\mathbf{w}$ refers to a contiguous sequence of natural language words that are immediately associated with the current semantic unit, while $\mathbf{X}$ and $\mathbf{Y}$ refers to a sequence of natural language words that the first and second child semantic unit will map to, respectively.

For example, in Figure 2 (b), the word sequence directly below the semantic unit m_a follows the pattern $\mathbf{wX}$ (since the word sequence $w_1w_2w_3$ is immediately associated with m_a, and the remaining words are remotely associated with m_a), and the word sequence below m_b follows $\mathbf{wXwYw}$[2].

The word association patterns are similar to those *hybrid patterns* used in hybrid trees. One key difference is that we disallow the unary pattern $\mathbf{X}$. The reason is, when computing the partition function in Equation 3, inclusion of pattern $\mathbf{X}$ will result in relaxed hybrid trees consisting of an infinite number of nodes. However, this issue does not come up in the original *hybrid tree* models due to their generative setting, where the training process does not involve such a partition function.

#Args	Word Association Patterns
0	$\mathbf{w}$
1	$\mathbf{wX}$, $\mathbf{Xw}$, $\mathbf{wXw}$
2	$\mathbf{XY}$, $\mathbf{YX}$, $\mathbf{wXY}$, $\mathbf{wYX}$, $\mathbf{XwY}$, $\mathbf{YwX}$ $\mathbf{XYw}$, $\mathbf{YXw}$, $\mathbf{wXwY}$, $\mathbf{wYwX}$ $\mathbf{wXYw}$, $\mathbf{wYXw}$, $\mathbf{XwYw}$, $\mathbf{YwXw}$ $\mathbf{wXwYw}$, $\mathbf{wYwXw}$

Table 2: The complete list of word association patterns. Here #Args means the number of arguments for a semantic unit.

3.4 Features

The features are defined over the $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ tuples. In practice, we define features at each level of the relaxed hybrid tree structure $\mathbf{h}$. In other words, features are defined over (n, m) tuples where n is a contiguous sequence of natural language words (immediately or remotely) associated with the semantic unit m (recall that $\mathbf{h}$ contains both $\mathbf{n}$ and $\mathbf{m}$, and each level of $\mathbf{h}$ simply consists of a semantic unit and a contiguous sequence of words). Each feature over (n, m) is then further decomposed as a product between two indicator feature functions, defined over the natural language words (n) and semantic unit (m) respectively: $\phi(n, m) = \phi^i(n) \times \phi^o(m)$. For each $\phi^i(n)$ we define two types of features: the *local features*, which are defined over immediately associated words only, and the *span features*, which are defined over all (immediately or remotely) associated words to capture long range dependencies.

The local features include word unigrams and bigrams, the word association patterns, as well as character-level features[3] which perform implicit morphological analysis. The span features include word unigrams, bigrams, as well as trigrams. Although our model allows certain more sophisticated features to be exploited, such as word POS features, word similarity features based on the WordNet (Pedersen et al., 2004), we deliberately choose to only include these simple features so

[2] This is based on the assumption that m_c and m_d are the first and second child of m_b in the semantic representation, respectively. If m_d is the first child in the semantic representation and m_c is the second, the pattern should be $\mathbf{wYwXw}$.

[3] For each word, we used all its prefixes (not necessarily linguistically meaningful ones) whose length are greater than 2 as features, for all languages.

as to make a fair comparison with previous works which also did not make use of external resources. For the features defined on m (i.e., $\phi^o(m)$), we include only the string form of m, as well as m's function name as features.

Finally, we also define features over $\mathbf{m}$ only. Such features are defined over semantic unit pairs such as (m_a, m_b) where m_a is the parent node of m_b as in $\mathbf{m}$. They include: 1) concatenation of the string forms of m_a and m_b, 2) concatenation of the string form of m_a and m_b's type, and 3) concatenation of the function names of m_a and m_b.

4 Algorithms

In this section we describe the efficient algorithms used for learning and decoding. The algorithms are inspired by the inside-outside style algorithms used for the generative hybrid tree models (Lu et al., 2008), but are different in the following ways: 1) we need to handle features, including long-distance features, 2) we need to additionally handle the computation of the partition function of Equation (3).

4.1 Learning

The training process involves the computation of the objective function (4) as well as the gradient terms (5).

The objective function (4) (excluding the regularization term which can be trivially computed) is equivalent to the following:

$$\mathcal{L}(\Lambda) = \sum_i \log \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}_i, \mathbf{m}_i)} e^{\Lambda \cdot \Phi(\mathbf{n}_i, \mathbf{m}_i, \mathbf{h})}$$
$$- \sum_i \log \sum_{\mathbf{m}', \mathbf{h}' \in \mathcal{H}(\mathbf{n}_i, \mathbf{m}')} e^{\Lambda \cdot \Phi(\mathbf{n}_i, \mathbf{m}', \mathbf{h}')} \quad (6)$$

In the first term, $\sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}_i, \mathbf{m}_i)} e^{\Lambda \cdot \Phi(\mathbf{n}_i, \mathbf{m}_i, \mathbf{h})}$ is in fact the sum of the scores (as defined by Φ and Λ) associated with all such latent structures that contain both $\mathbf{m}_i$ and $\mathbf{n}_i$ exactly. The second term is the sum of the scores associated with all the latent structures that contain $\mathbf{n}_i$ exactly. We focus our discussions on the computation of the first part first.

We use $\overset{m_{(p)}}{\underset{w_i \ldots w_j}{\triangle}}$ to denote the combined score of all such latent relaxed hybrid tree structures that contain both the semantic tree rooted by m and the natural language word sequence $w_i \ \ldots \ w_j$ that forms the word association pattern p with respect to m. For example, the score of the relaxed hybrid tree in Figure 2 (b) is contained by $\overset{m_{a(\mathbf{wX})}}{\underset{w_1 \ldots w_{10}}{\triangle}}$ (here $p = \mathbf{wX}$ because only $w_1 w_2 w_3$ are immediately associated with m_a).

We give an illustrative example that shows how these scores can be computed efficiently using dynamic programming. Consider the following case when m has at least one child semantic unit:

$$\overset{m_{(\mathbf{wXw})}}{\underset{w_i \ldots w_j}{\triangle}} = \overset{m_{(\mathbf{w})}}{\underset{w_i}{\triangle}} \otimes \overset{m_{(\mathbf{wXw})}}{\underset{w_{i+1} \ldots w_j}{\triangle}}$$
$$+ \overset{m_{(\mathbf{w})}}{\underset{w_i}{\triangle}} \otimes \overset{m_{(\mathbf{Xw})}}{\underset{w_{i+1} \ldots w_j}{\triangle}}$$

Here the symbol $\otimes$ means *extract and compute*, a process that involves 1) extraction of additional features when the two structures on the right-hand side are put together (for example, the local bigram feature "$w_i w_{i+1}$" can be extracted in the above case), and 2) computation of the score for the new structure when the two structures from both sides of $\otimes$ are combined, based on the scores of these structures and newly extracted features.

The above equation holds because for any relaxed hybrid tree contained by the left-hand side, the left-most word w_i is always immediately associated with m. The term $\overset{m_{(\mathbf{wXw})}}{\underset{w_{i+1} \ldots w_j}{\triangle}}$ is presenting a similar but smaller structure to the term on the left-hand side. The other term $\overset{m_{(\mathbf{wX})}}{\underset{w_i \ldots w_j}{\triangle}}$ can also be computed based on similar equations. In other words, such terms can be computed from even smaller similar terms in a recursive manner. A bottom-up dynamic programming algorithm is used for computing such terms.

When the semantic unit m has two child nodes, similar equations can also be established. Here we give an illustrative example:

$$\overset{m_{(\mathbf{wXwYw})}}{\underset{w_i \ldots w_j}{\triangle}} = \textstyle\sum_{k=i}^{j-1} \overset{m_{(\mathbf{wX})}}{\underset{w_i \ldots w_k}{\triangle}} \otimes \overset{m_{(\mathbf{wYw})}}{\underset{w_{k+1} \ldots w_j}{\triangle}}$$

Finally, we have the following equation:

$$\overset{m}{\underset{w_i \ldots w_j}{\triangle}} = \textstyle\sum_p \overset{m_{(p)}}{\underset{w_i \ldots w_j}{\triangle}}$$

The left-hand side simply means the combined score for all such relaxed hybrid trees that have (n, m) as the root, where $n \equiv w_i \ldots w_j$. Once the computation for a certain (n, m) pair is done, we can move up to process such pairs that involve m's parent node.

The above process essentially computes the *inside* score associated with the (n, m) pair, which gives the sum of the scores of all such (incomplete) relaxed hybrid trees that can be constructed with (n, m) as the root. Similar to (Lu et al., 2008), we can also define and compute the *outside* scores for (n, m) (the combined score of such incomplete relaxed hybrid trees that contain (n, m) as one of its leave nodes) in an analogous manner, where the computation of the gradient functions can be efficiently integrated in this process.

Computation of the second part of the objective function (6) involves dynamic programming over a packed forest representation rather than a single tree, which requires an extension to the algorithm described in (Lu et al., 2008). The resulting algorithm is similar to the one used in (Lu and Ng, 2011), which has been used for language generation from packed forest representations of typed λ-calculus expressions.

4.2 Decoding

The decoding phase involves finding the optimal semantic tree $\mathbf{m}^*$ given a new input sentence $\mathbf{n}$:

$$\mathbf{m}^* = \arg\max_{\mathbf{m}} P(\mathbf{m}|\mathbf{n}) \tag{7}$$

This in fact is equivalent to finding the following optimal semantic tree $\mathbf{m}^*$:

$$\mathbf{m}^* = \arg\max_{\mathbf{m}} \sum_{\mathbf{h}\in\mathcal{H}(\mathbf{n},\mathbf{m})} e^{\Lambda\cdot\Phi(\mathbf{n},\mathbf{m},\mathbf{h})} \tag{8}$$

Unfortunately, the summation operation inside the $\arg\max$ prevents us from employing a similar version of the dynamic programming algorithm we developed for learning in Section 4.1. To overcome this difficulty, we instead find the optimal semantic tree using the following equation:

$$\mathbf{m}^* = \arg\max_{\mathbf{m},\mathbf{h}\in\mathcal{H}(\mathbf{n},\mathbf{m})} e^{\Lambda\cdot\Phi(\mathbf{n},\mathbf{m},\mathbf{h})} \tag{9}$$

We essentially replace the $\sum$ operation by the $\max$ operation inside the $\arg\max$. In other words, we first find the best latent relaxed hybrid tree $\mathbf{h}^*$ that contains the input sentence $\mathbf{n}$, and next we extract the optimal semantic tree $\mathbf{m}^*$ from $\mathbf{h}^*$.

This decoding algorithm is similar to the dynamic programming algorithm used for computing the inside score for a given natural language sentence $\mathbf{n}$ (i.e., the algorithm for computing the second term of Equation (6)). The difference here is, at each intermediate step, instead of computing the combined score for all possible relaxed hybrid tree structures (i.e., performing `sum`), we find the single-best relaxed hybrid tree structure (i.e., performing $\max$).

5 Experiments

We present evaluations on the standard GeoQuery dataset which is publicly available. This dataset has been used for evaluations in various semantic parsing works (Wong and Mooney, 2006; Kate and Mooney, 2006; Lu et al., 2008; Jones et al., 2012). It consists of 880 natural language sentences paired with their corresponding formal semantic representations. Each semantic representation is a tree structured representation derived from a Prolog query that can be used to interact with a database of U.S. geography facts for retrieving answers. The original dataset was fully annotated in English, and recently Jones et al. (2012) released a new version of this dataset with three additional language annotations (German, Greek and Thai). For all the experiments, we used the identical experimental setup as described in Jones et al. (2012). Specifically, we trained on 600 instances, and evaluated on the remaining 280.

We note that there exist two different versions of the GeoQuery dataset annotated with completely different semantic representations. Besides the version that we use in this work, which is annotated with tree structured semantic representations, the other version is annotated with lambda calculus expressions (Zettlemoyer and Collins, 2005). Results obtained from these two versions are not comparable.[4] Like many previous works, we focus on tree structured semantic representations for evaluations in this work since our model is designed for handling the class of semantic representations with recursive tree structures.

We used the standard evaluation criteria for judging the correctness of the outputs. Specifically, our system constructs Prolog queries from the output parses, and uses such queries to retrieve answers from the GeoQuery database. An output is considered correct if and only if it retrieves the

[4] Kwiatkowski et al. (2010) showed in Table 3 of their work that the version with tree-structured representations appeared to be more challenging – their semantic parser's performance on this version was substantially lower than that on the lambda calculus version.

System	English		Thai		German		Greek	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
WASP	71.1	77.7	71.4	75.0	65.7	74.9	70.7	**78.6**
HYBRIDTREE+	76.8	81.0	73.6	76.7	62.1	68.5	69.3	74.6
UBL-S	82.1	82.1	66.4	66.4	**75.0**	**75.0**	73.6	73.7
TREETRANS	79.3	79.3	78.2	78.2	74.6	74.6	75.4	75.4
RHT (*all features*)	**83.6**	**83.6**	**79.3**	**79.3**	74.3	74.3	**78.2**	78.2

Table 3: Performance on the benchmark data, using four different languages as inputs. RHT: relaxed hybrid tree (this work).

same answers as the gold standard (Jones et al., 2012). We report accuracy scores – the percentage of inputs with correct answers, and F1 measures – the harmonic mean of precision (the proportion of correct answers out of inputs with an answer) and recall (the proportion of correct answers out of all inputs). By adopting such an evaluation method we will be able to directly compare our model's performance against those of the previous works.

The evaluations were conducted under such a setting in order to make comparisons to previous works. We would like to stress that our model is designed for general-purpose semantic parsing that is not only natural language-independent, but also task-independent. We thus distinguish our work from several previous works in the literature which focused on semantic parsing under other assumptions. Specifically, for example, works such as (Liang et al., 2013; Poon and Domingos, 2009; Clarke et al., 2010) essentially performed semantic parsing under different settings where the goal was to optimize the performance of certain downstream NLP tasks such as answering questions, and different semantic formalisms and language-specific features were usually involved.

For all our experiments, we used the L-BFGS algorithm for learning the feature weights, where feature weights were all initialized to zeros and the regularization hyper-parameter κ was set to 0.01. We set the maximum number of L-BFGS steps to 100. When all the features are considered, our model creates over 2 million features for each language on the dataset (English: 2.1M, Thai: 2.3M, German: 2.7M, Greek: 2.6M). Our model requires (on average) a per-instance learning time of 0.428 seconds and a per-instance decoding time of 0.235 seconds, on an Intel machine with a 2.2 GHz CPU. Our implementation is in Java. Here the per-instance learning time refers to the time spent on computing the instance-level log-likelihood as well as the expected feature counts (needed for the gradients).

Table 3 shows the evaluation results of our system as well as those of several other comparable previous works which share the same experimental setup as ours. UBL-S is the system presented in Kwiatkowski et al. (2010) which performs semantic parsing with the CCG based on mapping between graphs, and is the only non-tree based top-performing system. Their system, similar to ours, also uses a discriminative log-linear model where two types of features are defined. WASP is a model based on statistical phrase-based machine translation as we have described earlier. The hybrid tree model (HYBRIDTREE+) performs learning using a generative process which is augmented with an additional discriminative-reranking stage, where certain global features are incorporated (Lu et al., 2008). The Bayesian tree transducer model (TREETRANS) learns under a Bayesian generative framework, using hyper-parameters manually tuned on the German training data.

We can observe from Table 3 that the semantic parser based on relaxed hybrid tree gives competitive performance when all the features (described in Sec 3.4) are used. It significantly outperforms the hybrid tree model that is augmented with a discriminative reranking step. The model reports the best accuracy and F1 scores on English and Thai and best accuracy score on Greek. The scores on German are lower than those of UBL-S and TREETRANS, mainly because the span features appear not to be effective for this language, as we will discuss next.

We report in Table 4 the test set performance when certain types of features are excluded from our system. Such results can help us understand the effectiveness of features of different types. As we can see from the table, in general, all features play essential roles, though their effective-

System	English		Thai		German		Greek	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
RHT (*all features*)	**83.6**	**83.6**	79.3	79.3	74.3	74.3	78.2	78.2
RHT (*no local features*)	81.4	81.4	78.2	78.2	74.3	74.3	75.7	75.7
RHT (*no span features*)	81.1	81.1	77.9	77.9	**78.2**	**78.2**	**78.9**	**78.9**
RHT (*no char features*)	79.6	79.6	**82.1**	**82.1**	73.6	73.6	76.1	76.1

Table 4: Results when certain types of features (local features, span features and character-level features) are excluded.

ness vary across different languages. The local features, which capture local dependencies, are of particular importance. Performance on three languages (English, Thai, and Greek) will drop when such features are excluded. Character-level features are very helpful for the three European languages (English, German, and Greek), but appear to be harmful for Thai. This indicates the character-level features that we propose do not perform effective morphological analysis for this Asian language.[5] The span features, which are able to capture certain long-distance dependencies, also play important roles. Specifically, if such features are excluded, our model's performance on three languages (Greek, English, Thai) will drop. Such features do not appear to be helpful for Thai and appear to be harmful for German. Clearly, such long-distance features are not contributing useful information to the model when these two languages are considered. This is especially the case for German, where we believe such features are contributing substantial noisy information to the model. What underlying language-specific, syntactic properties are generally causing these gaps in the performances? We believe this is an important question that needs to be addressed in future research. As we have mentioned, to make an appropriate comparison with previous works, only simple features are used. We believe that our system's performance can be further improved when additional informative language-specific features can be extracted from effective language tools and incorporated into our system.

6 Conclusions

In this work, we present a new discriminative model for semantic parsing which extends the *hybrid tree* model. Such an extension is similar to the extension of the generative syntactic parser based on probabilistic context-free grammars (PCFG) to the feature-based CRF parser (Finkel et al., 2008), but is slightly more complex due to latent structures. Developed on top of our novel *relaxed hybrid tree* representations, our model allows certain long-distance dependencies to be captured. We also present efficient algorithms for learning and decoding. Experiments on benchmark data show that our model is competitive to previous works and achieves the state-of-the-art performance across several different languages.

Future works include development of efficient algorithms for feature-based semantic parsing with alternative loss functions (Zhou et al., 2013), development of feature-based language generation models (Lu et al., 2009; Lu and Ng, 2011) and multilingual semantic parsers (Jie and Lu, 2014), as well as the development of efficient semantic parsing algorithms for optimizing the performance of certain downstream NLP tasks with less supervision (Clarke et al., 2010; Liang et al., 2013).

Being able to efficiently exploit features defined over individual words, our model also opens up the possibility for us to exploit alternative representations of words for learning (Turian et al., 2010), or to perform joint learning of both distributional and logical semantics (Lewis and Steedman, 2013). Furthermore, as a general string-to-tree structured prediction model, this work may find applications in other areas within NLP.

The system and code can be downloaded from `http://statnlp.org/research/sp/`.

Acknoledgments

The author would like to thank the anonymous reviewers for their helpful comments. This work was supported by SUTD grant SRG ISTD 2013 064.

[5] The character-level features that we introduced are indeed very general. We have conducted several additional experiments, which show that our model's performance for each language can be further improved when certain language-specific character-level features are introduced.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228, June.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proc. of CONLL '10*, pages 18–27.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proc. of ACL/HLT*, pages 959–967.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proc. of CONLL '05*, pages 9–16.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proc. of ACL '11*, pages 1486–1495.

Zhanming Jie and Wei Lu. 2014. Multilingual semantic parsing: Parsing multiple languages into semantic representations. In *Proc. of COLING*, pages 1291–1301.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proc. of ACL '12*, pages 488–496.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proc. of COLING/ACL*, pages 913–920.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proc. EMNLP'10*, pages 1223–1233.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

D. C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, December.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proc. of EMNLP '11*, pages 1611–1622.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proc. of EMNLP '08*, pages 783–792.

Wei Lu, Hwee Tou Ng, and Wee Sun Lee. 2009. Natural language generation with tree conditional random fields. In *Proc. of EMNLP*, pages 400–409.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *Proc. of HLT-NAACL '04 (Demonstration)*, pages 38–41.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proc. of EMNLP '09*, pages 1–10.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Mark Steedman. 1996. *Surface structure and interpretation*. MIT press.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proc. of ACL '10*, pages 384–394.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. of HLT/NAACL '06*, pages 439–446.

Yuk Wah Wong and Raymond J Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proc. of ACL '07*.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI '05*.

Junsheng Zhou, Juhong Xu, and Weiguang Qu. 2013. Efficient latent structural perceptron with hybrid trees for semantic parsing. In *IJCAI*, pages 2246–2252.