Dependency-based Hybrid Trees for Semantic Parsing

Zhanming Jie and **Wei Lu**
Singapore University of Technology and Design
8 Somapah Road, Singapore, 487372
zhanming_jie@mymail.sutd.edu.sg, luwei@sutd.edu.sg

Abstract

We propose a novel *dependency-based hybrid tree* model for semantic parsing, which converts natural language utterance into machine interpretable meaning representations. Unlike previous state-of-the-art models, the semantic information is interpreted as the latent dependency between the natural language words in our joint representation. Such dependency information can capture the interactions between the semantics and natural language words. We integrate a neural component into our model and propose an efficient dynamic-programming algorithm to perform tractable inference. Through extensive experiments on the standard multilingual GeoQuery dataset with eight languages, we demonstrate that our proposed approach is able to achieve state-of-the-art performance across several languages. Analysis also justifies the effectiveness of using our new dependency-based representation.[1]

1 Introduction

Semantic parsing is a fundamental task within the field of natural language processing (NLP). Consider a natural language (NL) sentence and its corresponding meaning representation (MR) as illustrated in Figure 1. Semantic parsing aims to transform the natural language sentences into machine interpretable meaning representations automatically. The task has been popular for decades and keeps receiving significant attention from the NLP community. Various systems (Zelle and Mooney, 1996; Kate et al., 2005; Zettlemoyer and Collins, 2005; Liang et al., 2011) were proposed over the years to deal with different types of semantic representations. Such models include structure-based models (Wong and Mooney, 2006; Lu et al., 2008; Kwiatkowski et al., 2010; Jones et al., 2012) and neural network based models (Dong and Lapata, 2016; Cheng et al., 2017).

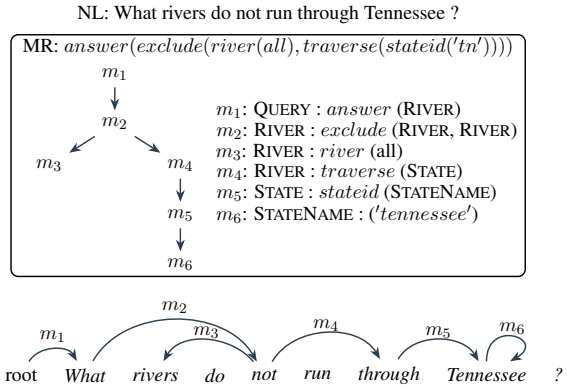

Figure 1: Top: natural language (NL) sentence; middle: meaning representation (MR); bottom: dependency-based hybrid tree representation.

Following various previous research efforts (Wong and Mooney, 2006; Lu et al., 2008; Jones et al., 2012), in this work, we adopt a popular class of semantic formalism – logical forms that can be equivalently represented as tree structures. The tree representation of an example MR is shown in the middle of Figure 1. One challenge associated with building a semantic parser is that the exact correspondence between the words and atomic semantic units are not explicitly given during the training phase. The key to the building of a successful semantic parsing model lies in the identification of a good *joint* latent representation of both the sentence and its corresponding semantics. Example joint representations proposed in the literature include a chart used in phrase-based translation (Wong and Mooney, 2006), a constituency tree-like representation known as *hybrid tree* (Lu et al., 2008), and a CCG-based derivation tree (Kwiatkowski et al., 2010).

Previous research efforts have shown the effec-

[1] We make our system and code available at http://statnlp.org/research/sp.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2431–2441
Brussels, Belgium, October 31 - November 4, 2018.

tiveness of using dependency structures to extract semantic representations (Debusmann et al., 2004; Cimiano, 2009; Bédaride and Gardent, 2011; Stanovsky et al., 2016). Recently, Reddy et al. (2016, 2017) proposed a model to construct logical representations from sentences that are parsed into dependency structures. Their work demonstrates the connection between the dependency structures of a sentence and its underlying semantics. Although their setup and objectives are different from ours where externally trained dependency parsers are assumed available and their system was trained to use the semantics for a specific down-stream task, the success of their work motivates us to propose a novel joint representation that can explicitly capture dependency structures among words for the semantic parsing task.

In this work, we propose a new joint representation for both semantics and words, presenting a new model for semantic parsing. Our main contributions can be summarized as follows:

- We present a novel *dependency-based hybrid tree* representation that captures both words and semantics in a joint manner. Such a dependency tree reveals semantic dependencies between words which are easily interpretable.
- We show that exact dynamic programming algorithms for inference can be designed on top of our new representation. We further show that the model can be integrated with neural networks for improved effectiveness.
- Extensive experiments conducted on the standard multilingual GeoQuery dataset show that our model outperforms the state-of-the-art models on 7 out of 8 languages. Further analysis confirms the effectiveness of our dependency-based representation.

To the best of our knowledge, this is the first work that models the semantics as latent dependencies between words for semantic parsing.

2 Related Work

The literature on semantic parsing has focused on various types of semantic formalisms. The λ-calculus expressions (Zettlemoyer and Collins, 2005) have been popular and widely used in semantic parsing tasks over recent years (Dong and Lapata, 2016; Gardner and Krishnamurthy, 2017; Reddy et al., 2016, 2017; Susanto and Lu, 2017a; Cheng et al., 2017). Dependency-based compositional semantics (DCS)[2] was introduced by Liang et al. (2011), whose extension, λ-DCS, was later proposed by Liang (2013). Various models (Berant et al., 2013; Wang et al., 2015; Jia and Liang, 2016) on semantic parsing with the λ-DCS formalism were proposed. In this work, we focus on the tree-structured semantic formalism which has been examined by various research efforts (Wong and Mooney, 2006; Kate and Mooney, 2006; Lu et al., 2008; Kwiatkowski et al., 2010; Jones et al., 2012; Lu, 2014; Zou and Lu, 2018).

Wong and Mooney (2006) proposed the WASP semantic parser that regards the task as a phrase-based machine translation problem. Lu et al. (2008) proposed a generative process to generate natural language words and semantic units in a joint model. The resulting representation is called *hybrid tree* where both natural language words and semantics are encoded into a joint representation. The UBL-s (Kwiatkowski et al., 2010) parser applied the CCG grammar (Steedman, 1996) to model the joint representation of both semantic units and contiguous word sequences which do not overlap with one another. Jones et al. (2012) applied a generative process with Bayesian tree transducer and their model also simultaneously generates the meaning representations and natural language words. Lu (2014, 2015) proposed a discriminative version of the hybrid tree model of (Lu et al., 2008) where richer features can be captured. Dong and Lapata (2016) proposed a sequence-to-tree model using recurrent neural networks where the decoder can branch out to produce tree structures. Susanto and Lu (2017b) augmented the discriminative hybrid tree model with multilayer perceptron and achieved state-of-the-art performance.

There exists another line of work that applies given syntactic dependency information to semantic parsing. Titov and Klementiev (2011) decomposed a syntactic dependency tree into fragments and modeled the semantics as relations between the fragments. Poon (2013) learned to derive semantic structures based on syntactic dependency trees predicted by the Stanford dependency parser. Reddy et al. (2016, 2017) proposed a linguistically motivated procedure to transform syntactic dependencies into logical forms. Their semantic parsing performance relies on the quality of the syntactic dependencies. Unlike such efforts, we do not re-

[2]Unlike ours, their work captures dependencies between semantic units but not natural language words.

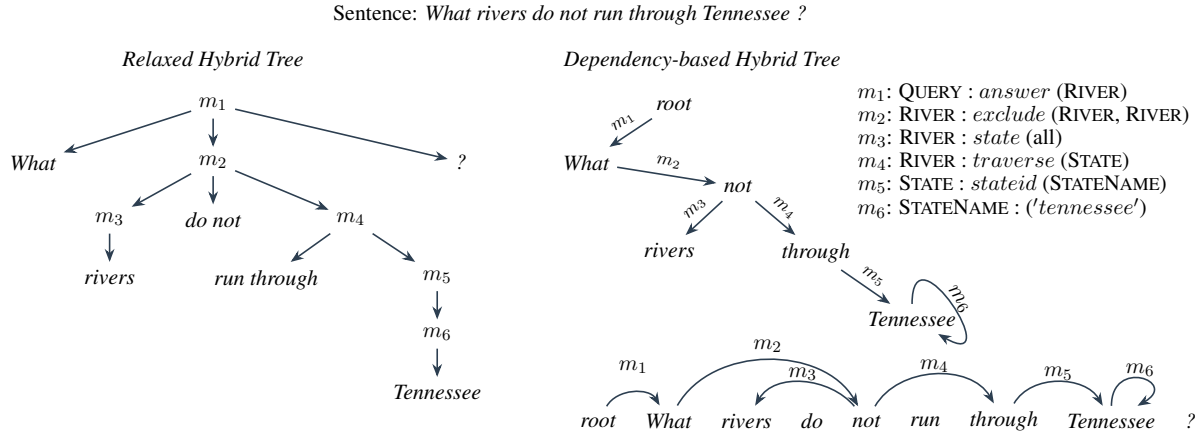

Figure 2: The *relaxed hybrid tree* (left) (Lu, 2014) and our *dependency-based hybrid tree* (right) as well as the flat representation (bottom right) of the example in Figure 1.

quire external syntactic dependencies, but model the semantic units as latent dependencies between natural language words.

3 Approach

3.1 Variable-free Semantics

The variable-free semantic representations in the form of FunQL (Kate et al., 2005) used by the de-facto GeoQuery dataset (Zelle and Mooney, 1996) encode semantic compositionality of the logical forms (Cheng et al., 2017). In the tree-structured semantic representations as illustrated in Figure 1, each tree node is a semantic unit of the following form:

$$m_i \equiv \tau_\alpha : p_\alpha(\tau^*_\beta)$$

where m_i denotes the complete semantic unit, which consists of semantic type τ_α, function symbol p_α and an argument list of semantic types τ^*_β (here $*$ denotes that there can be 0, 1, or 2 semantic types in the argument list. This number is known as the *arity* of m_i). Each semantic unit can be regarded as a function that takes in other (partial) semantic representations of certain types as arguments and returns a semantic representation of a specific type. For example in Figure 1, the root unit is represented by m_1, the type of this unit is QUERY, the function name is $answer$ and it has a single argument RIVER which is a semantic type. With recursive function composition, we can obtain a complete MR as shown in Figure 1.

3.2 Dependency-based Hybrid Trees

To jointly encode the tree-structured semantics $\boldsymbol{m}$ and a natural language sentence $\boldsymbol{n}$, we introduce our novel *dependency-based hybrid tree*. Figure 2 (right) shows the two equivalent ways of visualizing the dependency-based hybrid tree based on the example given in Figure 1. In this example, $\boldsymbol{m}$ is the tree-structured semantics $m_1(m_2(m_3, m_4(m_5(m_6))))$ and $\boldsymbol{n}$ is the sentence $\{w_1, w_2, \cdots, w_8\}$[3]. Our dependency-based hybrid tree $\boldsymbol{t}$ consists of a set of dependencies between the natural language words, each of which is labeled with a semantic unit. Formally, a dependency arc is represented as (w_p, w_c, m_i), where w_p is the *parent* of this dependency, w_c is the *child*, and m_i is the semantic unit that serves as the *label* for the dependency arc. A valid dependency-based hybrid tree (with respect to a given semantic representation) allows one to recover the correct semantics from it. Thus, one constraint is that for any two adjacent dependencies (w_p, w_c, m_i) and (w'_p, w'_c, m_j), where $w_c \equiv w'_p$, m_i must be the parent of m_j in the tree-structured representation $\boldsymbol{m}$. For example, in Figure 2, the dependencies (*not*, *through*, m_4) and (*through*, *Tennessee*, m_5) satisfy the above condition. However, we cannot replace (*through*, *Tennessee*, m_5) with, for example, (*through*, *Tennessee*, m_6), since m_6 is not the child of m_4. Furthermore, the number of children for a word in the dependency tree should be consistent with the arity of the corresponding semantic unit that points to it. For example, "*not*" has 2 children in our dependency-based hybrid tree representation because the semantic unit m_2 (i.e., RIVER : $exclude$ (RIVER, RIVER)) has arity 2. Also, "*rivers*" is the leaf as m_3, which points to it, has arity 0. We will discuss in Section 3.3

[3]We also introduce a special token "*root*" as w_0.

Abstract Semantic Unit	Arity	Dependency Pattern
A	0	**WW**
B	1	**X**, **WX**, **XW**
C	2	**XY**, **YX**

Table 1: List of dependency patterns.

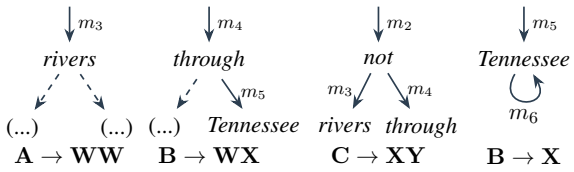

Figure 3: Example dependency patterns used in the dependency-based hybrid tree of Figure 2.

on how to derive the set of allowable dependency-based hybrid trees for a given $(\boldsymbol{m}, \boldsymbol{n})$ pair.

To understand the potential advantages of our new joint representation, we compare it with the *relaxed hybrid tree* representation (Lu, 2014), which is illustrated on the left of Figure 2. We highlight some similarities and differences between the two representations from the *span level* and *word level* perspectives.

In a relaxed hybrid tree representation, words and semantic units jointly form a constituency tree-like structure, where the former are leaves and the latter are internal nodes of such a joint representation. Such a representation is able to capture alignment between the natural language words and semantics at the span level.[4] For example, m_2 covers the span from "*rivers*" to "*Tennessee*", which allows the interactions between the semantic unit and the span to be captured. Similarly, in our dependency-based hybrid tree, such span level word-semantics correspondence can also be captured. For example, the arc between "*not*" and "*through*" is labeled by the semantic unit m_4. This also allows the interactions between m_4 and words within the span from "*not*" to "*through*" to be captured.

While both models are able to capture the span-level correspondence between words and semantics, we can observe that in the relaxed hybrid tree, some words within the span are more directly related to the semantic unit (e.g., "*do not*" are more related to m_2) and some are not. Specifically, in their representation, the span level information assigned to the parent semantic unit always contains the span level information assigned to all its child semantic units. This may not always be desirable and may lead to irrelevant features. In fact, Lu (2014) also empirically showed that the span-level features may not always be helpful in their representation. In contrast, in our dependency-based hybrid tree, the span covered by m_2 is from "*What*" to "*not*", which only consists of the span level information associated with its first child semantic units. Therefore, our representation is more flexible in capturing the correspondence between words and semantics at the span level, allowing the model to choose the relevant span for features.

Furthermore, our representation can also capture precise interactions between words through dependency arcs labeled with semantic units. For example, the semantic unit m_4 on the dependency arc from "*not*" to "*through*" in our representation can be used to capture their interactions. However, such information could not be straightforwardly captured in a relaxed hybrid tree, which is essentially a constituency tree-like representation. In the same example, consider the word "*not*" that bridges two arcs labeled by m_2 and m_4. Lexical features defined over such arcs can be used to indirectly capture the interactions between semantic units and guide the tree construction process. We believe such properties can be beneficial in practice, especially for certain languages. We will examine their significance in our experiments later.

3.3 Dependency Patterns

To define the set of allowable dependency-based hybrid tree representation so as to allow us to perform exact inference later, we introduce the *dependency patterns* as shown in Table 1. We use **A**, **B** or **C** to denote the abstract semantic units with arity 0, 1, and 2, respectively. We use **W** to denote a contiguous word span, and **X** and **Y** to denote the first and second child semantic unit, respectively.

We explain these patterns with concrete cases in Figure 3 based on the example in Figure 2. For the first case, the semantic unit m_3 has arity 0, the pattern involved is **WW**, indicating both the left-hand and right-hand sides of "*rivers*" (under the dependency arc with semantic unit m_3) are just word spans (**W**, whose length could be zero). In the second case, the semantic unit m_4 has arity 1, the pattern involved is **WX**, indicating the left-hand side of "*through*" (under the arc of semantic unit m_4) is a word span and the right-hand side should be handled by the first child of m_4 in the

[4] We refer readers to (Lu, 2014) for more details.

semantic tree, which is m_5 in this case. In the third case, the semantic unit m_2 has two arguments, and the pattern involved in the example is $\mathbf{XY}$, meaning the left-hand and right-hand sides should be handled by the first and second child semantic units (i.e., m_3 and m_4), respectively.[5] The final case illustrates that we also allow self-loops on our dependency-based hybrid trees, where an arc can be attached to a single word.[6] To avoid an infinite number of self-loops over a word, we set a maximum depth c to restrict the maximum number of recurrences, which is similar to the method introduced in (Lu, 2015).

Based on the dependency patterns, we are able to define the set of all possible allowable *dependency-based hybrid tree* representations. Each representation essentially belongs to a class of *projective* dependency trees where semantic units appear on the dependency arcs and (some of the) words are selected as nodes. The semantic tree can be constructed by following the arcs while referring to the dependency patterns involved.

3.4 Model

Given the natural language words $\boldsymbol{n}$, our task is to predict $\boldsymbol{m}$, which is a tree-structured meaning representation, consisting of a set of semantic units as the nodes in the semantic tree. We use $\boldsymbol{t}$ to denote a dependency-based hybrid tree (as shown in Figure 2), which jointly encodes both natural language words and the gold meaning representation. Let $\mathcal{T}(\boldsymbol{n}, \boldsymbol{m})$ denote all the possible dependency-based hybrid trees that contain the natural language words $\boldsymbol{n}$ and the meaning representation $\boldsymbol{m}$. We adopt the widely-used structured prediction model conditional random fields (CRF) (Lafferty et al., 2001). The probability of a possible meaning representation $\boldsymbol{m}$ and dependency-based hybrid tree $\boldsymbol{t}$ for a sentence $\boldsymbol{n}$ is given by:

$$P_{\mathbf{w}}(\boldsymbol{m}, \boldsymbol{t}|\boldsymbol{n}) = \frac{e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})}}{\sum_{\boldsymbol{m}',\boldsymbol{t}'\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m}')} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m}',\boldsymbol{t}')}}$$

where $\mathbf{f}(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$ is the feature vector defined over the $(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$ tuple, and $\mathbf{w}$ is the parameter vector. Since we do not have the knowledge of the "true" dependencies during training, $\boldsymbol{t}$ is regarded as a latent-variable in our model. We marginalize $\boldsymbol{t}$ in the above equation and the resulting model is a latent-variable CRF (Quattoni et al., 2005):

$$\begin{aligned} P_{\mathbf{w}}(\boldsymbol{m}|\boldsymbol{n}) &= \sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m})} P_{\mathbf{w}}(\boldsymbol{m}, \boldsymbol{t}|\boldsymbol{n}) \\ &= \frac{\sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m})} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})}}{\sum_{\boldsymbol{m}',\boldsymbol{t}'\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m}')} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m}',\boldsymbol{t}')}} \end{aligned} \quad (1)$$

Given a dataset $\mathcal{D}$ of $(\boldsymbol{n}, \boldsymbol{m})$ pairs, our objective is to minimize the negative log-likelihood:[7]

$$\mathcal{L}(\mathbf{w}) = -\sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \log \sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m})} P_{\mathbf{w}}(\boldsymbol{m}, \boldsymbol{t}|\boldsymbol{n}) \quad (2)$$

The gradient for model parameter w_k is:

$$\begin{aligned} \frac{\partial\mathcal{L}(\mathbf{w})}{\partial w_k} =& \sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \sum_{\boldsymbol{m}',\boldsymbol{t}} \mathbf{E}_{P_{\mathbf{w}}(\mathbf{m}',\boldsymbol{t}|\boldsymbol{n})}[f_k(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})] \\ &- \sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \sum_{\boldsymbol{t}} \mathbf{E}_{P_{\mathbf{w}}(\boldsymbol{t}|\boldsymbol{n},\boldsymbol{m})}[f_k(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})] \end{aligned}$$

where $f_k(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$ represents the number of occurrences of the k-th feature. With both the objective and gradient above, we can minimize the objective function with standard optimizers, such as L-BFGS (Liu and Nocedal, 1989) and stochastic gradient descent. Calculation of these expectations involves all possible dependency-based hybrid trees. As there are exponentially many such trees, an efficient inference procedure is required. We will present our efficient algorithm to perform exact inference for learning and decoding in the next section.

3.5 Learning and Decoding

We propose dynamic-programming algorithms to perform efficient and exact inference, which will be used for calculating the objective and gradients discussed in the previous section. The algorithms are inspired by the inside-outside style algorithm (Baker, 1979), graph-based dependency parsing (Eisner, 2000; Koo and Collins, 2010; Shi et al., 2017), and the *relaxed hybrid tree* model (Lu, 2014, 2015). As discussed in Section 3.3, our latent dependency trees are projective as in traditional dependency parsing (Eisner, 1996; Nivre and Scholz, 2004; McDonald et al., 2005) – the dependencies are non-crossing with respect to the word order (see bottom of Figure 1).

[5] Analogously, the pattern $\mathbf{YX}$ would mean m_4 handles the left-hand side and m_3 right-hand side.

[6] The limitations associated with disallowing such a pattern have been discussed in the previous work of (Lu, 2015).

[7] We ignore the L_2 regularization term for brevity.

The objective function in Equation 2 can be further decomposed into the following form[8]:

$$\mathcal{L}(\mathbf{w}) = -\sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \log \sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m})} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})} + \sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \log \sum_{\boldsymbol{m}',\boldsymbol{t}'\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m}')} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m}',\boldsymbol{t}')}$$

We can see the first term is essentially the combined score of all the possible latent structures containing the pair $(\boldsymbol{n}, \boldsymbol{m})$. The second term is the combined score for all the possible latent structures containing $\boldsymbol{n}$. We show how such scores can be calculated in a factorized manner, based on the fact that we can recursively decompose a dependency-based hybrid tree based on the dependency patterns we introduced.

Formally, we introduce two interrelated dynamic-programming structures that are similar to those used in graph-based dependency parsing (Eisner, 2000; Koo and Collins, 2010; Shi et al., 2017), namely *complete span* and *complete arc span*. Figure 4a shows an example of *complete span* (left) and *complete arc span* (right). The *complete span* (over $[i, j]$) consists of a headword (at i) and its descendants on one side (they altogether form a subtree), a dependency pattern and a semantic unit. The *complete arc span* is a span (over $[i, j]$) with a dependency between the headword (at i) and the modifier (at k). We use $C_{i,j,p,m}$ to denote a complete span, where i and j represent the indices of the headword and endpoint, p is the dependency pattern and m is the semantic unit. Analogously, we use $A_{i,k,j,p,m}$ to denote a complete arc span where i and k are used to denote the additional dependency from the word at the i-th position as headword to the word at the k-th position as modifier.

As we can see from the derivation in Figure 4, each type of span can be constructed from smaller spans in a bottom-up manner. Figure 4a shows that a complete span is constructed from a complete arc span following the dependency patterns in Table 1. Figure 4b shows a complete arc span can be simply constructed from two smaller complete spans based on the dependency pattern. In Figure 4c and 4d, we further show how such two complete spans with pattern $\mathbf{X}$ (or $\mathbf{Y}$) and $\mathbf{W}$ can be constructed. Figure 4c illustrates how to model a transition from one semantic unit to another where

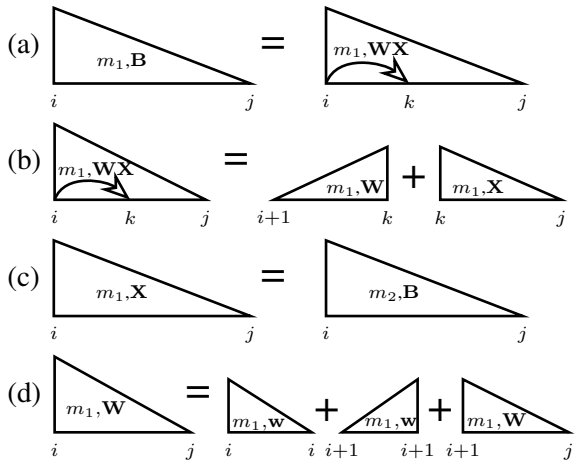

Figure 4: The dynamic-programing structures and derivation of our model. The other direction is symmetric. See supplementary material for the complete structures.

the parent is m_1 and the child is m_2 in the semantic tree. If m_2 has arity 1, then the pattern is $\mathbf{B}$ following the dependency patterns in Table 1. For spans with a single word, we use the lowercase $\mathbf{w}$ as the pattern to indicate this fact, as shown in Figure 4d. They are the atomic spans used for building larger spans. As the complete span in Figure 4d is associated with pattern $\mathbf{W}$, which means the words within this span are under the semantic unit m_1, we can incrementally construct this span with atomic spans. We illustrate the construction of a complete dependency-based hybrid tree in the supplementary material.

Our final goal during training for a sentence $\boldsymbol{n} = \{w_0, w_1, \cdots, w_N\}$ is to construct all the possible *complete spans* that cover the interval $[0, N]$, which can be represented as $C_{0,N,\cdot,\cdot}$. Similar to the chart-based dependency parsing algorithms (Eisner, 1996, 2000; Koo and Collins, 2010), we can obtain the *inside* and *outside* scores using our dynamic-programming derivation in Figure 4 during the inference process, which can then be used to calculate the objective and feature expectations. Since the spans are defined by at most three free indices, the dependency pattern and the semantic unit, our dynamic-programming algorithm requires $\mathcal{O}(N^3M)$ time[9] where M is the number of semantic units. The resulting complexity is the same as the relaxed hybrid tree model (Lu, 2014).

During decoding, we can find the optimal (tree-structured) meaning representation $\boldsymbol{m}^*$ for a given

[8] Regularization term is excluded for brevity.

[9] We omit a small constant factor associated with patterns.

Feature Type	Examples
Word	"m_4 & *run*", "m_4 & *through*"
Pattern	"m_2 & **XY**", "m_4 & **WX**"
Transition	"m_2 & m_3", "m_2 & m_4"
Head word	"m_2 & *What*", "m_4 & *not*"
Modifier word	"m_2 & *not*", "m_4 & *through*"
Bag of words	"m_4 & *not*", "m_4 & *run*", "m_4 & *through*"

Table 2: Features for the example in Figure 2.

input sentence $\boldsymbol{n}$ by the Viterbi algorithm. This step can also be done efficiently with our dynamic-programming approach, where we switch from marginal inference to MAP inference:

$$\boldsymbol{m}^*, \boldsymbol{t}^* = \underset{\boldsymbol{m},\boldsymbol{t} \in \mathcal{T}(\boldsymbol{n},\boldsymbol{m})}{\arg\max} \; e^{\mathbf{w} \cdot \mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})}$$

A similar decoding procedure has been used in previous work (Lu, 2014; Durrett and Klein, 2015) with CKY-based parsing algorithm.

3.6 Features

As shown in Equation 1, the features are defined on the tuple $(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$. With the dynamic-programming procedure, we can define the features over the structures in Figure 2. Our feature design is inspired by the hybrid tree model (Lu, 2015) and graph-based dependency parsing (McDonald et al., 2005). Table 2 shows the feature templates for the example in Figure 2. Specifically, we define simple unigram features (concatenation of a semantic unit and a word that directly appears under the unit), pattern features (concatenation of the semantic unit and the child pattern) and transition features (concatenation of the parent and child semantic units). They form our basic feature set.

Additionally, with the structured properties of dependencies, we can define dependency-related features (McDonald et al., 2005). We use the parent (head) and child (modifier) words of the dependency as features. We also use the bag-of-words covered under a dependency as features. The dependency features are useful in helping improve the performance as we can see in the experiments section.

3.7 Neural Component

Following the approach used in Susanto and Lu (2017b), we could further incorporate neural networks into our latent-variable graphical model. The integration is analogous to the approaches described in the neural CRF models (Do and Artieres, 2010; Durrett and Klein, 2015; Gormley, 2015; Lample et al., 2016), where we use neural networks to learn distributed feature representations within our graphical model.

We employ a neural architecture to calculate the score associated with each dependency arc (w_p, w_c, m) (here w_p and w_c are the parent and child words in the dependency and m is the semantic unit over the arc), where the input to the neural network consists of words (i.e., (w_p, w_c)) associated with this dependency and the neural network will calculate a score for each possible semantic unit, including m. The two words are first mapped to word embeddings $\mathbf{e}_p$ and $\mathbf{e}_c$ (both of dimension d). Next, we use a bilinear layer[10] (Socher et al., 2013; Chen et al., 2016) to capture the interaction between the parent and the child in a dependency:

$$r_i = \mathbf{e}_p^{\mathbf{T}} \mathbf{U}_i \mathbf{e}_c$$

where r_i represents the score for the i-th semantic unit and $\mathbf{U}_i \in \mathbb{R}^{d \times d}$. The scores are then incorporated into the probability expression in Equation 1 during learning and decoding. As a comparison, we also implemented a variant where our model directly takes in the average embedding of $\mathbf{e}_p$ and $\mathbf{e}_c$ as additional features, without using our neural component.

4 Experiments

Data and evaluation methodology We conduct experiments on the publicly available variable-free version of the GeoQuery dataset, which has been widely used for semantic parsing (Wong and Mooney, 2006; Lu et al., 2008; Jones et al., 2012). The dataset consists of 880 pairs of natural language sentences and the corresponding tree-structured semantic representations. This dataset is annotated with eight languages. The original annotation of this dataset is English (Zelle and Mooney, 1996) and Jones et al. (2012) annotated the dataset with three more languages: German, Greek and Thai. Lu and Ng (2011) released the Chinese annotation and Susanto and Lu (2017b) annotated the corpus with three additional languages: Indonesian, Swedish and Farsi. In order to compare with previous work (Jones et al., 2012; Lu, 2015), we follow the standard splits with 600 instances for training and 280 instances for testing. To evaluate the performance, we follow the

[10] Empirically, we also tried multilayer perceptron but the bilinear model gives us better results.

Type	System/Model	English (en)		Thai (th)		German (de)		Greek (el)		Chinese (zh)		Indonesian (id)		Swedish (sv)		Farsi (fa)	
		Acc.	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*
Non-Neural	WASP	71.1	77.7	71.4	75.0	65.7	74.9	70.7	78.6	48.2	51.6	74.6	79.8	63.9	71.5	46.8	54.1
	HYBRIDTREE	76.8	81.0	73.6	76.7	62.1	68.5	69.3	74.6	56.1	58.4	66.4	72.8	61.4	70.5	51.8	58.6
	UBL	82.1	82.1	66.4	66.4	75.0	75.0	73.6	73.7	63.8	63.8	73.8	73.8	78.1	78.1	64.4	64.4
	TREETRANS	79.3	79.3	78.2	78.2	74.6	74.6	75.4	75.4	-	-	-	-	-	-	-	-
	RHT	86.8	86.8	80.7	80.7	75.7	75.7	79.3	79.3	76.1	76.1	75.0	75.0	79.3	79.3	73.9	73.9
Neural	SEQ2TREE†	84.5	-	71.9	-	70.3	-	73.1	-	73.3	-	80.7	-	80.8	-	70.5	-
	MSP-SINGLE†	83.5	-	72.1	-	69.3	-	74.2	-	74.9	-	79.8	-	77.5	-	72.2	-
	NEURAL HT (J=0)	87.9	87.9	82.1	82.1	75.7	75.7	81.1	81.1	76.8	76.8	76.1	76.1	81.1	81.1	75.0	75.0
	NEURAL HT (J=1)	88.6	88.6	84.6	84.6	76.8	76.8	79.6	79.6	75.4	75.4	78.6	78.6	82.9	82.9	76.1	76.1
	NEURAL HT (J=2)	**90.0**	**90.0**	82.1	82.1	73.9	73.9	80.7	80.7	81.1	81.1	81.8	81.8	83.9	83.9	74.6	74.6
Non-Neural	(This work) DEPHT	86.8	86.8	81.8	81.8	76.1	76.1	80.4	80.4	81.4	81.4	86.8	86.8	85.4	85.4	73.9	73.9
Non-Neural	(This work) DEPHT + embedding	87.5	87.5	83.9	83.9	75.0	75.0	81.1	81.1	81.4	81.4	87.5	87.5	87.1	87.1	73.6	73.6
Neural	(This work) DEPHT + NN	89.3	89.3	**86.7**	**86.7**	**78.2**	**78.2**	**82.9**	**82.9**	**82.9**	**82.9**	**88.7**	**88.7**	**87.3**	**87.3**	**77.9**	**77.9**

Table 3: Performance comparison with state-of-the-art models on GeoQuery dataset. († represents the system is using lambda-calculus expressions as meaning representations.)

standard evaluation procedure used in various previous works (Wong and Mooney, 2006; Lu et al., 2008; Jones et al., 2012; Lu, 2015) to construct the Prolog query from the tree-structured semantic representation using a standard and publicly available script. The queries are then used to retrieve the answers from the GeoQuery database, and we report accuracy and F_1 scores.

Hyperparameters We set the maximum depth c of the semantic tree to 20, following Lu (2015). The L_2 regularization coefficient is tuned from 0.01 to 0.05 using 5-fold cross-validation on the training set. The Polyglot (Al-Rfou et al., 2013) multilingual word embeddings[11] (with 64 dimensions) are used for all languages. We use L-BFGS (Liu and Nocedal, 1989) to optimize the DEPHT model until convergence and stochastic gradient descent (SGD) with a learning rate of 0.05 to optimize the neural DEPHT model. We implemented our neural component with the Torch7 library (Collobert et al., 2011). Our complete implementation is based on the StatNLP[12] structured prediction framework (Lu, 2017).

4.1 Baseline Systems

We run the released systems of several state-of-the-art semantic parsers, namely the WASP parser (Wong and Mooney, 2006), HYBRIDTREE model (Lu et al., 2008), UBL system (Kwiatkowski et al., 2010), *relaxed hybrid tree* (RHT) (Lu, 2015)[13], the sequence-to-tree (SEQ2TREE) model (Dong and Lapata, 2016), the *neural hybrid tree* (NEURAL HT) model (Susanto and Lu, 2017b), and the multilingual semantic parser (Susanto and Lu, 2017a) with single language (MSP-SINGLE) as input. The results for TREETRANS (Jones et al., 2012) are taken from their paper.

[11]The embeddings are fixed to avoid overfitting.

[12]https://gitlab.com/sutd_nlp/statnlp-core

[13](Lu, 2015) is an extension of the original *relaxed hybrid tree* (Lu, 2014), which reports improved results.

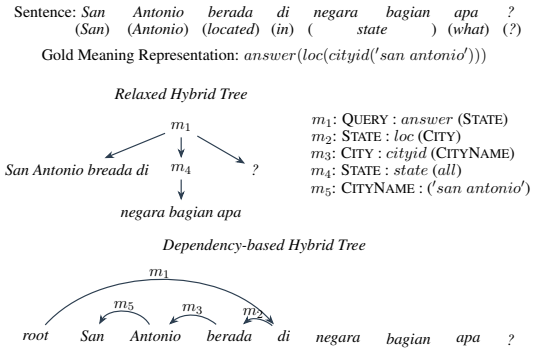

Figure 5: Example results from DEPHT and RHT on Indonesian.

4.2 Results and Discussion

Table 3 (top) shows the results of our dependency-based hybrid tree model compared with non-neural models which achieve state-of-the-art performance on the GeoQuery dataset. Our model DEPHT achieves competitive performance and outperforms the previous best system RHT on 6 languages. Improvements on the Indonesian dataset are particularly striking (+11.8 absolute points in F_1). We further investigated the outputs from both systems on Indonesian by doing error analysis. We found 40 instances that are incorrectly predicted by RHT are correctly predicted by DEPHT. We found that 77.5% of the errors are due to incorrect alignment between words and semantic units. Figure 5 shows an example of such errors where the relaxed hybrid tree fails to capture the correct alignment. We can see the question is asking "*What state is San Antonio located in?*". However, the natural language word order in Indone-

	en	th	de	el	zh	id	sv	fa
DEPHT basic	75.0	82.1	70.4	74.6	76.1	71.9	73.9	69.3
BASIC+HM feats.	80.7	**83.9**	**75.7**	79.2	81.1	85.0	81.1	72.5
BASIC+BOW feats.	**86.1**	83.2	73.9	**79.3**	**81.4**	**86.1**	**85.4**	**73.2**
DEPHT	86.8	81.8	76.1	80.4	81.4	86.8	85.4	73.9

Table 4: F_1 scores of our model with different dependency features.

sian is different from English, where the phrase "*berada di*" that corresponds to m_2 (i.e., loc) appears between "*San Antonio*" (which corresponds to m_5 – $'san\ antonio'$) and "*what*" (which corresponds to m_1 – $answer$). Such a structural non isomorphism issue between the sentence and the semantic tree makes the relaxed hybrid tree parser unable to produce a joint representation with valid word-semantics alignment. This issue makes the RHT model unable to predict the semantic unit m_2 (i.e., loc) as RHT has to align the words "*San Antonio*" which should be aligned to m_5 before aligning "*berada di*". However, m_5 has arity 0 and cannot have m_2 as its child. Thus, it would be impossible for the RHT model to predict such a meaning representation as output. In contrast, we can see that our dependency-based hybrid tree representation appears to be more flexible in handling such cases. The dependency between the two words "*di*" (*in*) and "*berada*" (*located*) is also well captured by the arc between them that is labeled with m_2. The error analysis reveals the flexibility of our joint representation in different languages in terms of the word ordering, indicating that the novel dependency-based joint representation is more robust and suffers less from language-specific characteristics associated with the data.

Effectiveness of dependency To investigate the helpfulness of the features defined over latent dependencies, we conduct ablation tests by removing the dependency-related features. Table 4 shows the performance of augmenting different dependency features in our DEPHT model with basic features. Specifically, we investigate the performance of head word and modifier word features (HM) and also the bag-of-words features (BOW) that can be extracted based on dependencies. It can be observed that dependency features associated with the words are crucial for all languages, especially the BOW features.

Effectiveness of neural component The bottom part of Table 3 shows the performance comparison among models that involve neural networks. Our DEPHT model with embeddings as features can outperform neural baselines across several languages (i.e., Chinese, Indonesian and Swedish). From the table, we can see the neural component is effective, which consistently gives better results than DEPHT and the approach that uses word embedding features only. Susanto and Lu (2017b) presented the NEURAL HT model with different window size J for their multilayer perceptron. Their performance will differ with different window sizes, which need to be tuned for each language. In our neural component, we do not require such a language-specific hyperparameter, yet our neural approach consistently achieves the highest performance on 7 out of 8 languages compared with all previous approaches. As both the embeddings and the neural component are defined on the dependency arcs, the superior results also reveal the effectiveness of our dependency-based hybrid tree representation.

5 Conclusions and Future Work

In this work, we present a novel *dependency-based hybrid tree* model for semantic parsing. The model captures the underlying semantic information of a sentence as latent dependencies between the natural language words. We develop an efficient algorithm for exact inference based on dynamic-programming. Extensive experiments on benchmark dataset across 8 different languages demonstrate the effectiveness of our newly proposed representation for semantic parsing.

Future work includes exploring alternative approaches such as transition-based methods (Nivre et al., 2006; Chen and Manning, 2014) for semantic parsing with latent dependencies, applying our dependency-based hybrid trees on other types of logical representations (e.g., lambda calculus expressions and SQL (Finegan-Dollak et al., 2018)) as well as multilingual semantic parsing (Jie and Lu, 2014; Susanto and Lu, 2017a).

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments on this work. We would also like to thank Yanyan Zou for helping us with running the experiments for baseline systems. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156, and is partially supported by project 61472191 under the National Natural Science Foundation of China.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of CoNLL*.

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

Paul Bédaride and Claire Gardent. 2011. Deep semantics for dependency structures. In *Proceedings of CICLing*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of ACL*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of ACL*.

Philipp Cimiano. 2009. Flexible semantic composition with dudes. In *Proceedings of ICCS*.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011. Torch7: A matlab-like environment for machine learning. In *Proceedings of BigLearn, NIPS workshop*.

Ralph Debusmann, Denys Duchier, Alexander Koller, Marco Kuhlmann, Gert Smolka, and Stefan Thater. 2004. A relational syntax-semantics interface based on dependency grammar. In *Proceedings of COLING*.

Trinh-Minh-Tri Do and Thierry Artieres. 2010. Neural conditional random fields. In *Proceedings of AISTAT*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of ACL-IJCNLP*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in probabilistic and other parsing technologies*, pages 29–61. Springer.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of ACL*.

Matt Gardner and Jayant Krishnamurthy. 2017. Open-vocabulary semantic parsing with both distributional statistics and formal knowledge. In *Proceedings of AAAI*.

Matthew R Gormley. 2015. *Graphical Models with Structured Factors, Neural Factors, and Approximation-Aware Training*. Ph.D. thesis.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

Zhanming Jie and Wei Lu. 2014. Multilingual semantic parsing: Parsing multiple languages into semantic representations. In *Proceedings of COLING*.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of ACL*.

Rohit J Kate and Raymond J Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of COLING/ACL*.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of AAAI*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of EMNLP*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *Technical Report arXiv*.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of NNACL*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.

Wei Lu. 2014. Semantic parsing with relaxed hybrid trees. In *Proceedings of EMNLP*.

Wei Lu. 2015. Constrained semantic forests for improved discriminative semantic parsing. In *Proceedings of ACL*.

Wei Lu. 2017. A unified framework for structured prediction: From theory to practice. In *Proceedings of EMNLP (Tutorial)*.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of EMNLP*.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of EMNLP*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of COLING*.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of ACL*.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2005. Conditional random fields for object recognition. In *Proceedings of NIPS*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of EMNLP*.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast (er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of EMNLP*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. Getting more out of syntax with props. *arXiv preprint arXiv:1603.01648*.

Mark Steedman. 1996. Surface structure and interpretation.

Raymond Hendy Susanto and Wei Lu. 2017a. Neural architectures for multilingual semantic parsing. In *Proceedings of ACL*.

Raymond Hendy Susanto and Wei Lu. 2017b. Semantic parsing with neural hybrid trees. In *Proceedings of AAAI*.

Ivan Titov and Alexandre Klementiev. 2011. A bayesian model for unsupervised semantic parsing. In *Proceedings of ACL*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of ACL-IJCNLP*, pages 1332–1342.

Yuk Wah Wong and Raymond J Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of NAACL*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of AAAI*.

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

Yanyan Zou and Wei Lu. 2018. Learning cross-lingual distributed logical representations for semantic parsing. In *Proceedings of ACL*.