Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Semantic Parsing with Neural Hybrid Trees

Raymond Hendy Susanto, Wei Lu
Singapore University of Technology and Design
8 Somapah Road, Singapore 487372
{raymond_susanto, luwei}@sutd.edu.sg

Abstract

We propose a neural graphical model for parsing natural language sentences into their logical representations. The graphical model is based on hybrid tree structures that jointly represent both sentences and semantics. Learning and decoding are done using efficient dynamic programming algorithms. The model is trained under a discriminative setting, which allows us to incorporate a rich set of features. Hybrid tree structures have shown to achieve state-of-the-art results on standard semantic parsing datasets. In this work, we propose a novel model that incorporates a rich, nonlinear featurization by a feedforward neural network. The error signals are computed with respect to the conditional random fields (CRFs) objective using an inside-outside algorithm, which are then backpropagated to the neural network. We demonstrate that by combining the strengths of the exact global inference in the hybrid tree models and the power of neural networks to extract high level features, our model is able to achieve new state-of-the-art results on standard benchmark datasets across different languages.

Introduction

Semantic parsing refers to the task of parsing natural language sentences into their corresponding semantic representations, such as the first-order logic or lambda calculus. It is one of the classical problems in natural language processing (NLP) and artificial intelligence. For example, consider the following natural language sentence paired with its corresponding semantic representation:

> Which states do not border Texas ?
> $answer(exclude(state(all), next\_to(stateid('tx'))))$

It is also possible to represent the above semantics as recursive structures such as trees, which consist of atomic semantic units as its nodes. For example, we can convert the above semantics into an equivalent tree structure as shown in Figure 1. Following previous works (Kate and Mooney 2006; Wong and Mooney 2006), we specifically focus on such tree-structured semantic representations.

A hybrid tree structure can represent a sentence-semantics pair as a tree where each node includes both natural language words and meaning representation tokens. However,

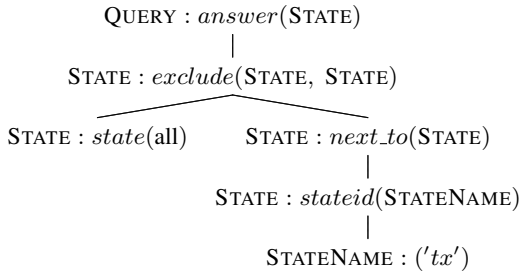

Figure 1: An example tree-structured semantic representation and its corresponding natural language sentence.

explicit labelings between words and meaning representation tokens are not available in the training data. This creates a latent variable learning problem that can be solved efficiently using an inside-outside algorithm (Lu et al. 2008). A *relaxed hybrid tree* (Lu 2014) is a discriminative version of the original hybrid tree model that is able to incorporate a wide range of features, some of which are able to capture long distance dependency. The *constrained semantic forest* addresses the under-specificity in the relaxed hybrid tree by limiting the height of the tree that should be considered (Lu 2015). Both extensions of the hybrid tree achieved state-of-the-art performance in semantic parsing.

Recently, neural networks have received a significant amount of interests from the NLP community. Neural approaches have shown its success in a variety of NLP tasks, ranging from parsing (Vinyals et al. 2015) to machine translation (Bahdanau, Cho, and Bengio 2015). One of the main advantages of using such neural approaches is its ability to learn a compact representation of its inputs, which often involves nonlinear interactions. It is common to treat an NLP task as a sequence-to-sequence problem where predictions are made sequentially using inexact inference, such as greedy or beam search. For example, the recent semantic parsing work in (Dong and Lapata 2016; Jia and Liang 2016) are based on sequential recurrent neural network models.

The idea of combining graphical models and neural networks has been explored in the past and started to see more interest in the recent years. For example, LeCun et al.

(1998) constructed graph transformer networks for recognizing character string, which combined convolutional neural networks with graph-based stochastic models. Collobert et al. (2011) proposed a model that optimizes sentence-level loglikelihood that incorporates neural network scores together with transition feature scores. Durrett and Klein (2015) used a neural CRF for constituency parsing based on the CKY algorithm. Our work integrates a neural network into a latent-variable model.

In this paper, we propose a neural approach to semantic parsing that takes structural information into account. Our approach builds upon the hybrid tree framework, where we use a feedforward neural network as a feature extractor. Both the hybrid tree model and the neural network are trained jointly using inside-outside and backpropagation, respectively. We evaluate our approach on multilingual benchmark datasets.

Related Work

Several approaches have been proposed to convert natural language sentences into logical forms. Examples include generative models (Lu et al. 2008; Jones, Johnson, and Goldwater 2012), discriminative models (Kate and Mooney 2006; Lu 2014), machine translation (Wong and Mooney 2006), and combinatory categorial grammars (CCG) (Zettlemoyer and Collins 2005; 2007; Kwiatkowski et al. 2010). From the literature, we highlight the following works that used the same evaluation dataset that allows meaningful comparison to their parsers' performance. For ease of reference, we give a name for each system (in **bold**).

Wong and Mooney (2006) proposed the **WASP** semantic parser based on statistical phrase-based machine translation where the word alignment model was used for lexical acquisition, and the parsing model was trained using the syntax-based translation model. The **UBL-S** parser (Kwiatkowski et al. 2010) used a probabilistic CCG grammar to represent the meaning of individual words and used higher-order unification to combine the meanings into a hypothesis space containing all grammatical structures consistent with the data it is trained on. Jones, Johnson and Goldwater (2012) introduced a variational Bayesian inference algorithm to build a generic tree transducer that can generate semantic parsing models, **TREETRANS**. Closer to our model, the original hybrid tree framework, **HYBRIDTREE+**, used a generative paradigm followed by a discriminative re-ranking stage (Lu et al. 2008). Thereafter, Lu (2014) proposed a discriminative version of the original hybrid tree, where some rigid dependency assumptions used in the generative version are now relaxed. In (Lu 2015), this model was further improved by introducing constrained semantic forest, **DISCHT+**.

Different from our model that adds neural features to a discriminative parsing framework, we highlight two prominent state-of-the-art neural approaches that used sequence generation methods. Dong et al. (2016) recently proposed a sequence-to-tree recurrent neural network (RNN) parser, **SEQ2TREE-DL**, and Jia and Liang (2016) trained a sequence-to-sequence RNN model with attention and copying mechanism on datapoints induced from an synchronous context-free grammar, **SEQ2SEQ-JL**.

Other works on semantic parsing make use of different semantic representations or perform evaluation under different settings. For example, a related work by (Andreas et al. 2016a) proposed a compositional model that learns to assemble neural network modules, instead of logical forms, for semantic parsing. They evaluated their approach on GeoQA, a geographical question-answering dataset introduced by Krishnamurthy and Kollar (2013), where a question is parsed into coarse modules with internal neural network machinery (Andreas et al. 2016b).

Model

We propose a *neural hybrid tree* model that consists of a graphical model[1] component, *i.e.*, a discriminative hybrid tree structure, and a multi-layer neural network. Our model is shown in Figure 2. We describe each component in the following sections.

Hybrid Trees

The *hybrid tree* model was first proposed as a generative semantic parsing framework in (Lu et al. 2008). Given a complete natural language sentence $\mathbf{n}$ and a complete semantic representation $\mathbf{m}$, we assume that there exists a complete latent structure $\mathbf{h}$ that jointly represents both $\mathbf{m}$ and $\mathbf{n}$. Essentially, this joint representation tells us the correct associations between words in $\mathbf{n}$ and semantics in $\mathbf{m}$.

Given the joint representations, a model can be built either generatively by modeling the joint probability distribution over $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ tuples, or discriminatively by modeling the conditional probability distribution over $(\mathbf{m}, \mathbf{h})$ pairs given $\mathbf{n}$. In this work, we focus on the latter approach. More formally, our model follows a log-linear approach:

$$P_\Lambda(\mathbf{m}, \mathbf{h}|\mathbf{n}) = \frac{\exp(F_\Lambda(\mathbf{n}, \mathbf{m}, \mathbf{h}))}{\sum_{\mathbf{m}', \mathbf{h}' \in \mathcal{H}(\mathbf{n}, \mathbf{m}')} \exp(F_\Lambda(\mathbf{n}, \mathbf{m}', \mathbf{h}'))} \quad (1)$$

$$F_\Lambda(\mathbf{n}, \mathbf{m}, \mathbf{h}) = \Lambda \cdot \Phi(\mathbf{n}, \mathbf{m}, \mathbf{h}) \quad (2)$$

where $\mathcal{H}(\mathbf{n}, \mathbf{m})$ is the set of all possible joint representations that contain both $\mathbf{n}$ and $\mathbf{m}$ and F is a linear scoring function that involves a discrete feature function Φ associated with a weight vector Λ. In practice, the latent structures are not observed. Hence, we consider the following marginal probability:

$$\begin{aligned} P_\Lambda(\mathbf{m}|\mathbf{n}) &= \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}, \mathbf{m})} P_\Lambda(\mathbf{m}, \mathbf{h}|\mathbf{n}) \\ &= \frac{\sum_{\mathbf{h} \in \mathcal{H}(\mathbf{n}, \mathbf{m})} \exp(F_\Lambda(\mathbf{n}, \mathbf{m}, \mathbf{h}))}{\sum_{\mathbf{m}', \mathbf{h}' \in \mathcal{H}(\mathbf{n}, \mathbf{m}')} \exp(F_\Lambda(\mathbf{n}, \mathbf{m}', \mathbf{h}'))} \end{aligned} \quad (3)$$

To limit the search space of the latent structures, we assume that $\mathbf{h}$ must be from a space consisting of hybrid tree structures with *relaxed* constraints (Lu 2014), which allow some long distance dependencies to be captured.

[1] Strictly speaking, the hybrid trees are parsing models, which are different from conventional graphical models as they involve hyperedges. We refer them as graphical models as both share similar inference procedures.

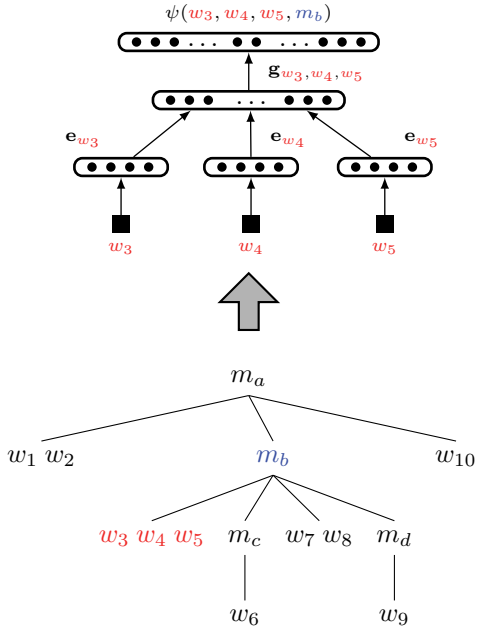

Figure 2: A neural hybrid tree

Figure 2 (bottom) shows an example of a hybrid tree encoding the sentence w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9 w_{10} and the semantics $m_a(m_b(m_c, m_d))$. In the generative hybrid tree, each word is strictly associated with a semantic unit. For example, the word w_3 is associated with the semantic unit m_b. In contrast, each word in the relaxed hybrid tree is not only directly associated with a semantic unit m, but also indirectly associated with all semantic units that are predecessors of m. Thus, the word w_3 is now directly associated with m_b, but is also indirectly associated with m_a.

Word Association Patterns At each level of the latent structure, we define *word association patterns* to specify how the words and child semantic unit are organized. The set of allowed patterns is: $\{\mathbf{w}, [\mathbf{w}]\mathbf{X}[\mathbf{w}], [\mathbf{w}]\mathbf{X}[\mathbf{w}]\mathbf{Y}[\mathbf{w}], [\mathbf{w}]\mathbf{Y}[\mathbf{w}]\mathbf{X}[\mathbf{w}]\}$, where $[\mathbf{w}]$ denotes an optional sequence of words. For example, in Figure 2, the word sequence directly below the semantic unit m_a follows the pattern $\mathbf{wXw}$, where we first have a word sequence that are directly associated with m_a (*i.e.*, $w_1 w_2$), followed by some words covered by its first child semantic unit, then another word sequence directly associated with m_a (*i.e.*, w_{10}). One important consideration here is the inclusion of pattern $\mathbf{X}$, which may lead to possible hybrid tree representations consisting of an infinite number of internal nodes (semantic units). In order to address such an issue, Lu (2015) added a constraint that limits the height of the semantic representation to a fixed constant c, where c is typically larger than the maximum height of all the trees appearing in the training set.

Features We define discrete features over the $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ tuples, *i.e.*, $\Phi(\mathbf{n}, \mathbf{m}, \mathbf{h})$ returns a vector consisting of *discrete* counts of features associated with the tuple. These features include emission features (concatenation of a semantic unit and an individual word that appears directly below it in the hybrid tree), pattern features (concatenation of a semantic unit and the pattern below the unit), and transition features (concatenation of two semantic units that form a parent-child relationship). Here in this work, we only used the above simple features. We will show in the later section that using these features in conjunction with neural features can lead to better performance.

Multi-layer Neural Networks

One important component in developing an end-to-end NLP system is *feature engineering*. We can enrich this process by including a neural network that takes raw inputs (*e.g.*, characters or words), which requires a little pre-processing effort. A compact representation of the inputs can be learned through the multi-layer structure of the network. In this section, we discuss the architecture of the neural network that is used in our semantic parser. Our design is influenced by that of (Collobert et al. 2011), a feedforward network that works efficiently in several benchmark NLP tasks, such as named entity recognition and semantic role labeling. We use a window approach, where we consider a fixed size J window of words around the target word that we want to tag.

Input Layer The first layer in the network accepts as input a word sequence $w_1, w_2, \ldots, w_n$. We define a *lookup table* layer that maps every word in the vocabulary to a d-dimensional vector, where $d \ll |\mathcal{V}|$. More formally, the word representation of w_i is given by $\mathbf{E}_{*,w_i}$, where $\mathbf{E} \in \mathbb{R}^{d \times |\mathcal{V}|}$ is a lookup table and $\mathbf{E}_{*,k}$ corresponds to the k-th column of $\mathbf{E}$. The representation of a word sequence is a simple concatenation of the embeddings of each word.

Leveraging the availability of large-scale monolingual corpora, we can initialize the lookup table with pre-trained word embeddings such as GloVe (Pennington, Socher, and Manning 2014) or Polyglot (Al-Rfou, Perozzi, and Skiena 2013). The lookup table can be fine-tuned according to a specific task during the backpropagation training.

Hidden Layer Given a d-dimensional input embedding $\mathbf{e}$, each hidden layer l applies an affine transformation:

$$\mathbf{g}^{(l)} = \sigma(\mathbf{W}^{(l)}\mathbf{g}^{(l-1)} + \mathbf{b}^{(l)}) \tag{4}$$

where $\mathbf{g}^{(0)} = \mathbf{e}$. $\mathbf{W}^{(1)} \in \mathbb{R}^{h \times d}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{h \times h}$ $(l = 2, \ldots, L-1)$, and $\mathbf{b}^{(l)} \in \mathbb{R}^h$ are weight matrices and bias vectors. Moreover, we use an element-wise nonlinear activation σ. Common choices of σ include hyperbolic tangent (tanh) and rectified linear unit (ReLU).

Output Layer The last layer L does a final affine transformation without nonlinear activation:

$$\mathbf{g}^{(L)} = \mathbf{W}^{(L)}\mathbf{g}^{(L-1)} + \mathbf{b}^{(L)} \tag{5}$$

The number of output nodes in this layer is equal to the number of labels in the task of interest, *i.e.*, $\mathbf{W}^{(L)} \in \mathbb{R}^{o \times h}$. For example, o is equal to the number of possible semantic units in our case. This gives us a vector of scores for each label, which is then used in the objective function computations

(*e.g.*, softmax, or in this work, the CRFs objective). Essentially, the network learns a function ψ that scores a given word sequence $w_1, w_2, ..., w_n$ and a semantic unit m_j:

$$\psi(w_1, ..., w_n, m_j) = \mathbf{g}_j^{(L)} \quad (6)$$

Neural Hybrid Trees

We propose a novel, integrated framework that is a natural marriage between a nonlinear featurization by using a neural network and a discriminative, latent-variable model that relies on hybrid tree structures within an efficient structured inference framework.

The neural network is utilized as a nonlinear feature extractor, in addition to the original discrete *non-neural* feature function. There are several benefits of using such neural features. First, we are able to induce a dense representation of features that are made of a wider context information, thanks to the underlying multi-layer architecture of the network. If these inputs were used as discrete features, it would be prone to the data sparseness issue which often results in overfitting. Second, unlike discrete word representations, where each word is treated independently among each other, our input layer consists of a low-dimensional lookup table that does not have such an independence assumption. To some extent, it improves the robustness of the model.

We combine the hybrid tree structures and multi-layer neural networks in a principled way. We call such a model *neural hybrid trees*. More specifically, it is straightforward to incorporate the neural scores (Equation 6) into our scoring function (Equation 2). The new scoring function is thus defined as follows:

$$F_{\Lambda,\Theta}(\mathbf{n}, \mathbf{m}, \mathbf{h}) = \Lambda \cdot \Phi(\mathbf{n}, \mathbf{m}, \mathbf{h}) + G_\Theta(\mathbf{n}, \mathbf{m}, \mathbf{h}) \quad (7)$$

where Θ is the parameters of the neural networks and G is a neural scoring function over the $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ tuples. In this work, we define features at each level of the hybrid tree $\mathbf{h}$. That is, the neural features are defined over pairs of the window surrounding a word w_j in $\mathbf{n}$ and the semantic unit m immediately associated with w_j. We denote an input window $w_{j-J}, ..., w_j, ..., w_{j+J}$ as ω_j. Thus, we define the neural scoring function as follows:

$$G(\mathbf{n}, \mathbf{m}, \mathbf{h}) = \sum_{(\omega,m)\in\mathcal{W}(\mathbf{n},\mathbf{m},\mathbf{h})} c(\omega, m, \mathbf{n}, \mathbf{m}, \mathbf{h}) \times \psi(\omega, m) \quad (8)$$

where $\mathcal{W}(\mathbf{n}, \mathbf{m}, \mathbf{h})$ is the set of (ω, m) pairs that can be extracted from $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ and c returns their number of occurences. We include start-of-sentence, end-of-sentence, and padding symbols to make the length of the word sequence (the window size) constant.

As an illustration, the neural network in Figure 2 takes a target word, a preceeding word, and a following word as input, *i.e.*, we have the window size $J = 1$. We look at the word w_4, which is directly attached to the semantic unit m_b. By feeding w_4 and its context to the neural network, we obtain the score for the tuple (w_3, w_4, w_5, m_b), *i.e.*, $\psi(w_3, w_4, w_5, m_b)$. In the same manner, we can compute all such features in a particular hybrid tuple $(\mathbf{n}, \mathbf{m}, \mathbf{h})$ to obtain the value of $G(\mathbf{n}, \mathbf{m}, \mathbf{h})$.

Learning and Decoding

The training process involves the computation of the objective function and the gradient terms. We define the log-likelihood objective for the training set as:

$$\mathcal{L}(\Lambda, \Theta) = \sum_i \log \sum_{\mathbf{h}\in\mathcal{H}(\mathbf{n}_i,\mathbf{m}_i)} \exp(F_{\Lambda,\Theta}(\mathbf{n}_i, \mathbf{m}_i, \mathbf{h})) - \sum_i \log \sum_{\mathbf{m}',\mathbf{h}'\in\mathcal{H}(\mathbf{n}_i,\mathbf{m}')} \exp(F_{\Lambda,\Theta}(\mathbf{n}_i, \mathbf{m}', \mathbf{h}')) \quad (9)$$

The first term computes the potential of $(\mathbf{n}, \mathbf{m})$. The second term is a normalization term. In both cases, we need to perform a summation over all possible latent hybrid tree structures. This can be done using a bottom-up dynamic programming approach as described in (Lu 2014). The approach essentially computes the *inside* score associated with an (n, m) pair, which gives the sum of scores of all such hybrid trees rooted at (n, m). The computation of the second term involves dynamic programming over a packed forest representation rather than a single tree.

Our goal is to maximize this objective function by tuning the model parameters, *i.e.*, the discrete feature weight vector Λ and neural network parameters Θ. We first consider computing the gradient for Λ. Assume that $\Lambda = \langle\lambda_1, \lambda_2, \ldots, \lambda_N\rangle$. Differentiating with respect to λ_k, the weight associated with the k-th discrete feature ϕ_k, yields:

$$\frac{\partial\mathcal{L}(\Lambda, \Theta)}{\partial\lambda_k} = \sum_i \sum_{\mathbf{h}} \mathbf{E}_{P_{\Lambda,\Theta}(\mathbf{h}|\mathbf{n}_i,\mathbf{m}_i)}[\phi_k(\mathbf{n}_i, \mathbf{m}_i, \mathbf{h})] - \sum_i \sum_{\mathbf{m},\mathbf{h}} \mathbf{E}_{P_{\Lambda,\Theta}(\mathbf{m},\mathbf{h}|\mathbf{n}_i)}[\phi_k(\mathbf{n}_i, \mathbf{m}, \mathbf{h})] \quad (10)$$

Analogous to the inside score computation, we can define and compute *outside* scores for (n, m), from which the above gradient can be computed efficiently.

Next, we need to compute the gradient for the neural network parameters Θ. The idea here is to provide the error signals for the output layer of the neural network. Recall that each node in the output layer computes a score associated with a semantic unit m and an input window ω. Hence, we first compute:

$$\frac{\partial\mathcal{L}(\Lambda, \Theta)}{\partial\psi(\omega, m)} = \sum_i \sum_{\mathbf{h}} \mathbf{E}_{P_{\Lambda,\Theta}(\mathbf{h}|\mathbf{n}_i,\mathbf{m}_i)}[c(\omega, m, \mathbf{n}_i, \mathbf{m}_i, \mathbf{h})] - \sum_i \sum_{\mathbf{m},\mathbf{h}} \mathbf{E}_{P_{\Lambda,\Theta}(\mathbf{m},\mathbf{h}|\mathbf{n}_i)}[c(\omega, m, \mathbf{n}_i, \mathbf{m}, \mathbf{h})] \quad (11)$$

The above gradients can be computed using the same dynamic programming principle. Intuitively, we treat the neural features in a similar way as the discrete non-neural features. However, instead of having a separate weight for each neural feature (essentially each (ω, m) pair), the weights are computed using a feedforward neural network. Thus, their gradients can be computed in the same manner as computing

the gradients for the discrete features. Subsequently, we pass these gradients to the neural network and apply the chain rule to compute gradients for Θ.

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

$$\begin{aligned}\mathbf{m}^* &= \arg\max_{\mathbf{m}} P(\mathbf{m}|\mathbf{n}) \\ &= \arg\max_{\mathbf{m}} \sum_{\mathbf{h}\in\mathcal{H}(\mathbf{n},\mathbf{m})} \exp(F_{\Lambda,\Theta}(\mathbf{n},\mathbf{m},\mathbf{h})) \quad (12)\end{aligned}$$

The computation of $P(\mathbf{m}|\mathbf{n})$ in Equation 12 above involves a summation over all possible hybrid tree structures. To make the computation efficient using dynamic programming, we replace the $\sum$ operation above with max, resulting in the following equation:

$$\mathbf{m}^* = \arg\max_{\mathbf{m},\mathbf{h}\in\mathcal{H}(\mathbf{n},\mathbf{m})} \exp(F_{\Lambda,\Theta}(\mathbf{n},\mathbf{m},\mathbf{h})) \quad (13)$$

Then, we use a Viterbi decoding algorithm to find the best latent hybrid tree $\mathbf{h}^*$, from which we can extract the optimal semantic tree $\mathbf{m}^*$.

Experiments and Results

Datasets and Evaluation

We evaluate our approach on the multilingual GeoQuery dataset, which is a standard benchmark evaluation for semantic parsing (Wong and Mooney 2006; Kate and Mooney 2006; Lu et al. 2008; Jones, Johnson, and Goldwater 2012). The data consists of 880 pairs of natural language sentences and corresponding semantic representations. In this work, our model is designed for handling the class of tree-structured semantic representations.

The initial version of GeoQuery is in English. Further releases include Chinese (Lu and Ng 2011), German, Greek, and Thai (Jones, Johnson, and Goldwater 2012). In this work, we annotated the corpus with additional three languages: Indonesian, Swedish, and Farsi. Each version was annotated by a native speaker of the respective language. We use the standard train/test split (600/280) in order to make our results comparable to previous works.

Following previous works (Jones, Johnson, and Goldwater 2012; Lu 2014), our evaluation uses a standard script that constructs Prolog queries based on the system outputs. The queries are then used to retrieve answers from the GeoQuery database. An output is correct if and only if the retrieved answer is the same as the gold standard. We report the accuracy and F_1-measure percentage.

Implementations and Settings

The hybrid tree framework was implemented in Java. The original implementation is publicly available. We follow the experimental settings in (Lu 2015). In particular, we use the L-BFGS algorithm (Liu and Nocedal 1989) with a maximum of 100 iterations. We set c, the maximum height of a semantic tree, to 20. Discrete (non-neural) feature weights are ℓ_2-regularized with a weight of 0.01.

We implement the neural networks using Torch7 library (Collobert, Kavukcuoglu, and Farabet 2011). The network is trained using backpropagation. The parameters are updated for 100 iterations using gradient descent with Adam updates (Ba and Kingma 2015), where the initial learning rate is 0.001. We use the Polyglot word embeddings (Al-Rfou, Perozzi, and Skiena 2013), which provides pre-trained models for several languages. The dimension of the word embedding is 64. We wrote a generic interface to facilitate interaction between the hybrid tree framework and the neural network back-end through socket communication.[2]

Our hyperparameter tuning for the neural network includes the choice of the activation function {tanh, ReLU}, the number of hidden units {50,100,150,200}, the number of hidden layers {0,1,2}, and the amount of dropout regularization {0,0.25,0.5}. We select these parameters through validation on the English dataset by further splitting the training set into 400 instances for training and 200 instances for tuning. Our final selection is the following: tanh activation, 100 hidden units, 1 hidden layer, and no dropout. Another hyperparameter that is worth noting is the window size J {0,1,2}. We deem that this hyperparameter is more language-specific, hence, we report evaluation results on all different window sizes.

The original hybrid tree model is non-convex due to latent variables. Adding nonlinear features through neural networks increases the model's complexity. Therefore, to help learning, we divide our training procedure into two phases, where each phase focuses on a certain region of the feature space. In the first phase, we focus on optimizing the non-neural feature weights by training the original hybrid tree model in (Lu 2015), which does not utilize a neural network. We then use the trained weights to initialize the non-neural weight vector Λ. The second phase is the neural hybrid tree training, where we optimize the neural parameters Θ. We found empirically during our development that setting these weights to fixed values yields better results.[3] We make our system, code and our newly created datasets on three languages available at http://www.statnlp.org/research/sp/.

Results

Table 1 shows evaluation results of our system as well as other systems from previous works under the same experimental settings. Particularly, we compare our system with different window sizes {0,1,2} against WASP, HYBRIDTREE+, UBL-S, TREETRANS, DISCHT+, SEQ2TREE-DL and SEQ2SEQ-JL[4] systems.

Results show that our system consistently obtains higher results than all the previous systems, including the discriminative hybrid tree model that does not have any neural features. We further evaluate our system on four language datasets, which have not been tried by previous works. Table 2 shows consistent improvements over DISCHT+. The

[2] http://zeromq.org/

[3] Our implementation allows fine-tuning these weights jointly with the neural parameters.

[4] SEQ2TREE-DL and SEQ2SEQ-JL reported the evaluation results on English only. SEQ2TREE-DL evaluated based on exact match using the lambda calculus representation. See Section 4.3 of (Dong and Lapata 2016).

		English		Thai	
		Acc.	F	Acc.	F
WASP		71.1	77.7	71.4	75.0
HYBRIDTREE+		76.8	81.0	73.6	76.7
UBL-S		82.1	82.1	66.4	66.4
TREETRANS		79.3	79.3	78.2	78.2
DISCHT+		86.8	86.8	80.7	80.7
SEQ2TREE-DL		87.1	-	-	-
SEQ2SEQ-JL		89.3	-	-	-
This work	$J = 0$	87.9	87.9	82.1	82.1
	$J = 1$	88.6	88.6	**84.6**	**84.6**
	$J = 2$	**90.0**	**90.0**	82.1	82.1
		German		Greek	
WASP		65.7	74.9	70.7	78.6
HYBRIDTREE+		62.1	68.5	69.3	74.6
UBL-S		75.0	75.0	73.6	73.7
TREETRANS		74.6	74.6	75.4	75.4
DISCHT+		75.7	75.7	79.3	79.3
This work	$J = 0$	75.7	75.7	**81.1**	**81.1**
	$J = 1$	**76.8**	**76.8**	79.6	79.6
	$J = 2$	73.9	73.9	80.7	80.7

Table 1: Performance of various works across four different languages. Acc.: accuracy %, F: F_1-measure %.

		Chinese		Indonesian	
		Acc.	F	Acc.	F
DISCHT+		76.1	76.1	75.0	75.0
This work	$J = 0$	76.8	76.8	76.1	76.1
	$J = 1$	75.4	75.4	78.6	78.6
	$J = 2$	**81.1**	**81.1**	**81.8**	**81.8**
		Swedish		Farsi	
DISCHT+		79.3	79.3	73.9	73.9
This work	$J = 0$	81.1	81.1	75.0	75.0
	$J = 1$	82.9	82.9	**76.1**	**76.1**
	$J = 2$	**83.9**	**83.9**	74.6	74.6

Table 2: Performance across additional four languages evaluated in this work. Acc.: accuracy %, F: F_1-measure %.

best result for each dataset is written in **bold**. Except for Greek, the best result for each language is obtained with a window size $J > 0$. Nonetheless, we observe that there is a consistent improvement over DISCHT+ even with $J = 0$, which essentially replaces sparse word representation with a dense embedding. For German, we notice a performance drop when using $J = 2$. This is in line with the finding reported in (Lu 2014), where including longer distance features appears to be harmful for German.

Discussions

Model Robustness

We evaluate the robustness of the model on a synthetic dataset automatically created from the original corpus. Specifically, we use WordNet (Fellbaum 1998) to replace word occurrences in the original test split with their synonyms. First, we start with the English dataset and run a part-of-speech tagger (Bird, Klein, and Loper 2009) to obtain all verbs, nouns, and adjectives. We replace words that occur in their lemma form with a random synonym obtained from the first sense in WordNet. Stop words are ignored. In the next step, we generate the synonym corpus for the remaining 7 languages by substituting words that are replaced in the English version. We utilize the extended version of the Open Multilingual WordNet (Bond and Foster 2013).

Table 3 compares the performance of DISCHT+ and our model. We use the best performing J for each language. First, we observe that replacing words with synonyms results in a performance drop. This is expected because this process often generates out-of-vocabulary (OOV) words. The number of OOV words also depends on the WordNet size for each language. On average, the absolute drop in performance (in terms of F_1 measure) is 1.6% for DISCHT+, while for our model the drop is 1.2%. This shows that our model, which makes use of word embeddings as features, degrades robustly in the presence of OOV. Moreover, the same conclusion still holds, that is, our neural hybrid tree model outperforms the discriminative hybrid tree without neural features.

	DISCHT+		This work	
	F	diff.	F	diff.
English	84.3	-2.5	89.6	-0.4
Thai	78.9	-1.8	83.2	-1.4
German	75.7	0.0	76.8	0.0
Greek	78.6	-0.7	80.4	-0.7
Chinese	71.8	-4.3	76.4	-4.7
Indonesian	73.9	-1.1	81.4	-0.4
Swedish	77.5	-1.8	82.1	-1.8
Farsi	73.2	-0.7	75.7	-0.4
avg. diff.		-1.6		-1.2

Table 3: Performance on the synonym datasets. F: F_1-measure %, diff.: difference % from original F_1.

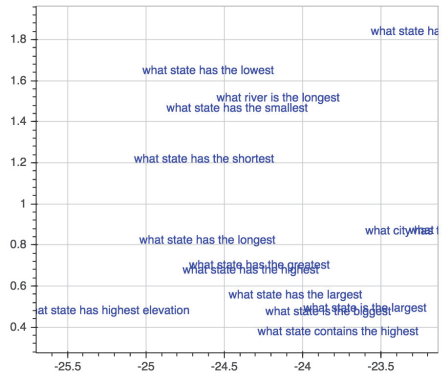

Figure 3: 2-D projection of learned phrase representations

Learned Window Representation

We visualize the representation learned by the hidden layer of the neural network. Since the input to the network is a window of consecutive words, essentially the hidden layer

is learning an embedding for each phrase, which potentially carries semantically meaningful information. In the figure, we plot the hidden layer output for English phrases ($J = 2$) found in GeoQuery. The vector is projected into a 2-dimensional space using Barnes-Hut-SNE (Van Der Maaten 2013). Each representation is a 100-dimensional vector. It can be observed that syntactically similar phrases are grouped together (*what state has the ...*). Moreover, semantically similar words (*highest*, *greatest*) are grouped together and more separated from their antonyms (*lowest*, *smallest*).

Conclusion

This paper presents a neural hybrid tree framework for semantic parsing. Our model integrates two different, yet complementing paradigms: graphical models and neural networks. We showed through empirical evaluations that our approach is competitive to the state-of-the-art in semantic parsing for multiple languages. It would be interesting to devise a unified approach that jointly transforms natural language sentences from multiple languages into a shared semantic representation. Potentially, we can improve our neural model with a recurrent architecture, such as long short-term memory. We leave these for future work.

Acknowledgments

We would also like to thank the anonymous reviewers for their helpful comments. This work is supported by MOE Tier 1 grant SUTDT12015008, and is partially supported by project 61472191 under the National Natural Science Foundation of China.

References

Al-Rfou, R.; Perozzi, B.; and Skiena, S. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*.

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016a. Learning to compose neural networks for question answering. In *Proceedings of NAACL*.

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016b. Neural module networks. In *Proceedings of CVPR*.

Ba, J., and Kingma, D. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

Bond, F., and Foster, R. 2013. Linking and extending an open multilingual Wordnet. In *Proceedings of ACL*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Collobert, R.; Kavukcuoglu, K.; and Farabet, C. 2011. Torch7: A Matlab-like environment for machine learning. In *Proceedings of BigLearn, NIPS Workshop*.

Dong, L., and Lapata, M. 2016. Language to logical form with neural attention. In *Proceedings of ACL*.

Durrett, G., and Klein, D. 2015. Neural CRF parsing. In *Proceedings of ACL-IJCNLP*.

Fellbaum, C. 1998. *WordNet*. Wiley Online Library.

Jia, R., and Liang, P. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

Jones, B. K.; Johnson, M.; and Goldwater, S. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of ACL*.

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

Krishnamurthy, J., and Kollar, T. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics* 1:193–206.

Kwiatkowski, T.; Zettlemoyer, L.; Goldwater, S.; and Steedman, M. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of EMNLP*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Liu, D. C., and Nocedal, J. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* 45(1-3):503–528.

Lu, W., and Ng, H. T. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of EMNLP*.

Lu, W.; Ng, H. T.; Lee, W. S.; and Zettlemoyer, L. S. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of EMNLP*.

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

Lu, W. 2015. Constrained semantic forests for improved discriminative semantic parsing. In *Proceedings of ACL-IJCNLP*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*.

Van Der Maaten, L. 2013. Barnes-Hut-SNE. In *Proceedings of ICLR*.

Vinyals, O.; Kaiser, Ł.; Koo, T.; Petrov, S.; Sutskever, I.; and Hinton, G. 2015. Grammar as a foreign language. In *Proceedings of NIPS*.

Wong, Y. W., and Mooney, R. J. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of NAACL*.

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

Zettlemoyer, L. S., and Collins, M. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*.