JOINT SEMANTIC UTTERANCE CLASSIFICATION AND SLOT FILLING WITH RECURSIVE NEURAL NETWORKS

Daniel (Zhaohan) Guo, Gokhan Tur, Wen-tau Yih, Geoffrey Zweig

Microsoft Research

ABSTRACT

In recent years, continuous space models have proven to be highly effective at language processing tasks ranging from paraphrase detection to language modeling. These models are distinctive in their ability to achieve generalization through continuous space representations, and compositionality through arithmetic operations on those representations. Examples of such models include feed-forward and recurrent neural network language models. Recursive neural networks (RecNNs) extend this framework by providing an elegant mechanism for incorporating *both* discrete syntactic structure *and* continuous-space word and phrase representations into a powerful compositional model. In this paper, we show that RecNNs can be used to perform the core spoken language understanding (SLU) tasks in a spoken dialog system, more specifically domain and intent determination, concurrently with slot filling, in one jointly trained model. We find that a very simple RecNN model achieves competitive performance on the benchmark ATIS task, as well as on a Microsoft Cortana conversational understanding task.

***Index Terms*—** Recursive Neural Networks, Dialog Systems, Domain Classification, Intent Determination, Slot Filling

1. INTRODUCTION

Recursive neural networks (RecNNs) have had a long history of application, being a natural and effective way of merging recursively structured inputs and continuous space representations [1, 2, 3, 4, 5, 6, 7]. These models operate on the parse structure of a sentence, and associate a continuous space vector with each leaf or word in the parse tree, and with each internal node. Such a vector can be thought of as encapsulating a semantic representation of the subtree rooted beneath it. The vectors are recursively computed from the leaves up, using a simple function to combine the vectors of child nodes into the vector for a root node. The model is trained so that the internal vectors are effective at semantic classification tasks, such as determining the polarity of the sentiment expressed by the words in a subtree, or in our case, to predict user intent.

The basic structure is illustrated in Figure 1. A parse tree is shown on top of a sentence, with the internal nodes labeled by their syntactic types, and the continuous-space vector associated with each node shown above the syntactic type. The details of this model are discussed in Section 3. Recently, recursive neural network models have demonstrated outstanding performance in tasks such as sentiment classification and paraphrase detection [7, 6], and even image analysis [5].

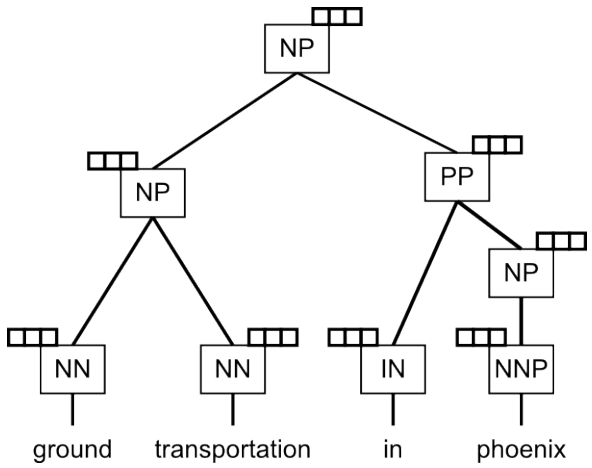

Fig. 1. Basic RecNN Structure. Internal vectors are represented with small boxes, e.g. 3 dimensional vectors in this case.

Motivated by the inherent ability of recursive neural networks to fuse discrete syntactic information with continuous space representations, in this paper we show how to apply them to spoken dialog processing, and in particular, we present a recursive network model to jointly perform the SLU tasks of domain detection, intent determination, and slot filling. The main contributions of this paper are:

- Adaptation of the recursive neural network to perform slot filling;
- The joint training of the recursive neural network for domain detection, intent determination, and slot filling; and
- An analysis of the appropriate level of syntactic tying for these tasks.

Table 1. ATIS utterance example
Domain: Airline Travel
Intent: Flight Information

Sentence	Slot label
show	O
flights	O
from	O
boston	B-fromloc.city_name
to	O
new	B-toloc.city_name
york	I-toloc.city_name
today	B-date

The remainder of the paper is organized as follows. Section 2 provides background on the spoken dialog tasks. Section 3 describes the recursive neural network model in detail, and our modifications to make it work for these tasks. Section 4 contains experimental results. Finally we make concluding remarks in Section 5.

2. SEMANTIC UTTERANCE CLASSIFICATION AND SLOT FILLING

Spoken language understanding in human/machine spoken dialog systems aims to automatically identify the domain and intent of the user as expressed in natural language (semantic utterance classification), and to extract associated arguments (slot filling). An example is shown in Table 1, which uses the IOB representation, where an initial "B" in a label indicates the beginning of a slot and an "I" indicates an extension of it. "O" is the null label. Once intent and slots are identified, the system can then decide on the next proper action to take according to the domain specific semantic template.

While these tasks are intimately related, traditionally domain/intent determination and slot filling have been done separately. Domain detection and intent determination tasks aim to classify a given speech utterance x into one of M semantic classes, $\hat{c}_i$, based on the contents of the utterance:

$$\hat{c}_i = \underset{i \in M}{\operatorname{argmax}}\, p(c_i|x) \tag{1}$$

To this end, a number of standard classifiers can be used, such as support vector machines and boosting [8, 9].

Slot filling, on the other hand, is usually framed as a sequence classification task. Formally, the task is to find most probable slot assignments $\hat{y}$ given utterance x:

$$\hat{y} = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}}\, p(y|x) \tag{2}$$

where $\mathcal{Y}(x)$ is the entire search space of slot assignments of x. For slot filling, conditional random field (CRF) [10] is a proven technique and has been used extensively [11], but recently recurrent neural networks (RNNs) have also shown excellent performance [12, 13, 14].

Recently, there has been some initial work on *joint* intent and slot determination. One approach used triangular CRF [15], which coupled an additional random variable for intent on top of a standard CRF. Mairesse et al. [16] used SVM classifiers to parse inputs, and apply semantic labels the interior nodes, resulting in both slot and intent labels. Another approach improved the model by using a convolutional neural network combined with a CRF [17]. In this paper, we extend this line of research with recursive neural networks.

3. RECURSIVE NEURAL NETWORKS FOR DIALOG SYSTEMS

Recursive neural networks extend traditional neural networks, enabling them to apply to structured tree inputs. This is done by repeatedly applying a neural network at each node of the tree to combine the output vectors of its children to form the node's own output vector. Recurrent Neural Networks (RNNs) can be thought of as a limiting case of recursive neural networks, where RNNs repeatedly apply a neural network to a degenerate tree (a chain) that has no notion of syntactic types. In previous work, RecNNs have been used for parsing natural language sentences [4, 5], as well as sentiment detection [7], and paraphrase detection [6]. Compared to other models that have been applied to these tasks, RecNNs are able to naturally take advantage of the syntactic tree structure in the input, augmented with vectors for compositional semantic information. Despite the success, RecNNs, unlike recurrent neural networks, have not been used for SLU tasks.

In this work, we present an adaptation of RecNNs to enable joint domain, intent and slot classification. As shown in Figure 1, the basic model structure goes as follows. We assume that each utterance is associated with a constituency parse tree, where the leaves of the tree correspond to the words of the sentence. Every word is associated with a word vector, and these vectors are given as input to the bottom of the network. Then the network propagates the information upwards by repeatedly applying a neural network at each node until the root node outputs a single vector. This vector is then used as the input to a semantic classifier, and the network is trained via backpropagation to maximize the performance of this classifier. Because the same neural network is recursively applied, the dimensions of the outputs of all the nodes must match.

3.1. Dialog-Specific Specializations

To tackle the tasks described in Section 2, several extensions and modifications are necessary.

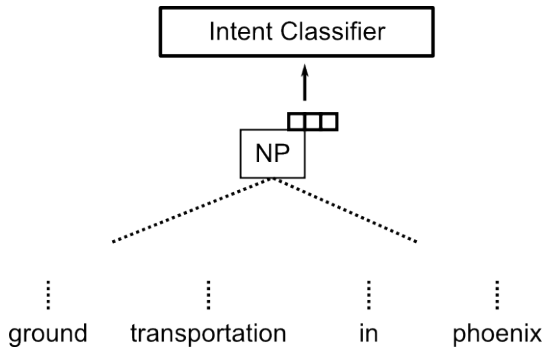

Fig. 2. The intent classifier

3.1.1. Domain and Intent Classification

Since the intents are domain specific, predicting the labels of domains and intents can be viewed as a single classification problem. Instead of training two separate models, we apply our RecNNs model to intent classification directly. The domain is then implied by the resulting intent. The setting we choose is similar to the standard multi-class maximum entropy model. The dot product of the output vector at the root of the tree and a vector for every possible intent is first computed. Then the posterior over the intent labels is derived by taking the softmax function of these dot products. This process is schematically illustrated in Figure 2.

3.1.2. Modeling Context for Slot Classification

If the slots of the sentences were to match up exactly with the span of subtrees in the parse tree, then a natural fit would be to put a classifier at every node in the tree, predicting the slot label of the subtrees. However, this is not the case for over 10% of the data, which consists of cases where the smallest subtree that spans a slot either contains another slot or contains many more words that are not part of the slot. Therefore, we take the traditional approach of token-wise classification for each individual word.

In order to bring more context information into the slot classifier, we also take advantage of the tree structure by adding tree-derived features. To do this, we take the output vectors of the nodes along the path up from the leaf to the root, and multiply each by a weight vector that is associated with the syntactic type of the node, and sum them up to get a path vector:

$$z = w^{(t_1)} * x_1 + w^{(t_2)} * x_2 + ... + w^{(t_k)} * x_k \qquad (3)$$

where $*$ is element-wise multiplication. Conceptually, Eq. (3) captures the contextual information that spans up to the whole sentence by aggregating these path vectors associated with every leaf node. To include more contextual information, we concatenate the previous and the next path vectors with the current path vector to form a tri-path vector (Figure 3).

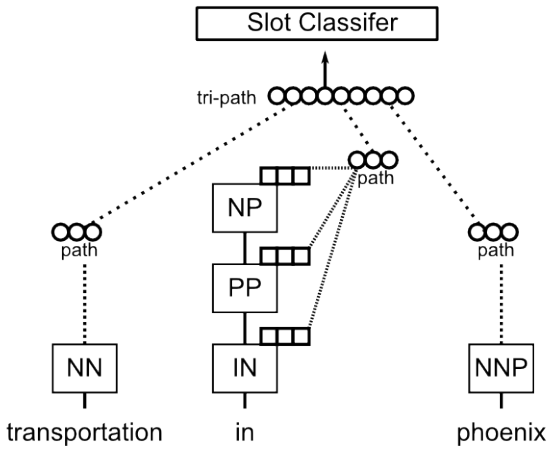

Fig. 3. Tri-path Features. The slot classifier is applied to each word in turn. The small circles are the aggregated path vectors.

Each tri-path vector is used as input to a multi-class maximum entropy classifier to predict the IOB slot label of the corresponding word. During training, the error signal is back-propagated, and used to adjust all matrices and atomic word representations involved.

3.2. Other Modifications

Apart from the SLU specific changes, we have also incorporated a few other general modifications to improve the model.

3.2.1. Multi-way Branches

The parse trees that most parsers output do not have a fixed branching factor or arity. In the simple case that the parse tree is a binary tree, then the neural network to be applied at each node can be defined as [4]:

$$y = tanh(W[x_1; x_2] + b) \qquad (4)$$

where $y, x_1, x_2, b \in \mathbb{R}^{n\times 1}$, $[x_1; x_2] \in \mathbb{R}^{2n\times 1}$ is the concatenation of the two output vectors from the children, and $W \in \mathbb{R}^{n\times 2n}$. The $tanh$ applies the hyperbolic tangent function element-wise. Alternatively, this can also be defined as

$$y = tanh(W_1 x_1 + W_2 x_2 + b) \qquad (5)$$

where $W_1, W_2 \in \mathbb{R}^{n\times n}$, and $W = [W_1, W_2]$ is the concatenation side by side. Using this formulation, we can extend the model to handle variable arity trees, by applying a position-dependent W to each child. This extension, also used in [18] is much more natural than turning them into binary trees, which results in unnatural splits as well as artificially deeper trees.

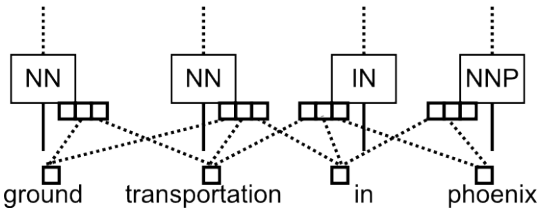

Fig. 4. Trigram Leaves

3.2.2. Syntactic Tying

To further use the syntactic information at each node of the tree, instead of using the same neural network at every node, or position-dependent variants, we can have a different neural network for every syntactic type i.e. a $(W^{(t)}, b^{(t)})$ for each syntactic type t (e.g. NP, PP, etc ...):

$$y = tanh(W^{(t)}x_1 + W^{(t)}x_2 + \cdots + W^{(t)}x_k + b^{(t)}) \quad (6)$$

So child vectors are combined differently depending on the syntactic type of the current node. This technique has been previous explored in [4, 19], which significantly improved the results.

3.2.3. Low-Rank Matrix Approximations

To speed up training and testing time, as well as a form of regularization, we use low-rank approximations to the $W^{(t)}$ matrices: $W^{(t)} = U^{(t)}V^{(t)} + D^{(t)}$ where $U^{(t)} \in \mathbb{R}^{n\times r}$, $V^{(t)} \in \mathbb{R}^{r\times n}$, and $D^{(t)} \in \mathbb{R}^{n\times n}$ is diagonal, and r is the rank. We picked $r = 1$ for our model.

3.2.4. Trigram Leaves

Traditional neural networks can be extended to take n-grams as input, improving performance due to more contextual information. Our RecNN can also be modified to naturally incorporate n-gram inputs. At the leaves of the tree, the word vectors for the preceding and proceeding words can be concatenated together to form a trigram, to be propagated up (Figure 4). In this case, the dimension of the vectors in the tree are three times the dimension of the word vectors.

3.3. Training

We use the sum of the cross-entropy error of the softmax classifiers as the objective function. We use mini-batches of size 50, and RMSProp to train the RecNN models [20]. RMSProp scales the individual weight updates by an estimate of the root-mean-square value of the recent gradient values. We found it beneficial to gradually reduce the overall learning rate by 0.9995 at each update. We use L2 regularization to help prevent over-fitting. All parameters except for the biases (the b's) are randomly initialized (including the word vectors). We took out 15% of the training data to use as hold-out data to determine the L2 regularization and which iteration for early stopping. After optimizing for these hyper-parameters, we then train the model on all of the training data.

3.4. Viterbi Sequence Optimization for Slot Filling

One big advantage of CRF models to RecNNs is that, it is performing global sequence optimization using tag level features followed by the well-known Viterbi optimization. In order to approximate this behavior, and optimize the sentence level tag sequence, we explicitly applied the Viterbi algorithm after fully training the RecNN model. To this end, a second order Markov model has been formed, using the slot tags, $t_i \in \mathcal{T}$ as states, where the state transition probabilities, $P(t_i|t_j)$ are obtained using a trigram tag language model (LM). The tag level posterior probabilities obtained from the RecNN model are used when computing the state observation likelihoods.

$$\hat{T} = \underset{T}{\operatorname{argmax}}\, P(T|W) \sim \underset{T}{\operatorname{argmax}}\, P_{LM}(T)^W \times P(W|T)$$
$$\sim \underset{T}{\operatorname{argmax}}\, P_{LM}(T)^W \times (\prod_i P_{RecNN}(t_i|w_i)/P(t_i))$$

As is often done in the speech community, when combining probabilistic models of different types, it is advantageous to weigh the contribution of the language model. We do so by introducing a tunable model combination weight (W), whose value is optimized on held-out data. For computation, we used the SRILM toolkit[1].

4. EXPERIMENTAL RESULTS

We evaluated the RecNN models on the ATIS database [21, 22]. We also include previous results of intent classification and slot labeling for comparison. The ATIS data consists of sentences of people making flight reservations. The there are 4978 training sentences, and 893 test sentences. The number of distinct intent labels is 18, and the number of distinct slot labels is 127, including the null label.

The ATIS dataset also has extra features termed "named-entity features," that give semantic labels for words in slots that nearly determine the slot label. We chose not to use these extra features as they are hand-crafted features that are not generally available. We have not included results from models that use these hand-crafted features. Instead, our models use only the words in a sentence as the input.

As can be seen from Table 2, RecNN performs comparably in intent classification to other state-of-the-art models. As for slot filling, RecNN (with Viterbi) most notably achieves better results than the popular CRF approach, and rivals the recurrent neural network (RNN) and the CNN+TriCRF[2].

[1] http://www.speech.sri.com/projects/srilm/

[2] The CNN+TriCRF model does do joint intent and slot filling, however they only have results for a different variation of the ATIS data, and their ac-

Table 2. Comparison to previous approaches. The 95% significance level for intent classification is ±1.4%; and ±0.8% for slot filling.

Model	Intent	Slot (F1)
SVM [11]	–	89.76
CRF [23]	–	91.09
RNN [12]	–	94.11
CNN+TriCRF [17]	–	94.35
Boosting [21]	95.72	–
Boosting + Simplified sentences [22]	96.98	–
RecNN (this work)	95.40	93.22
RecNN+Viterbi (this work)	95.40	93.96

Table 3. Performance of different syntactic tying schemes

Type	Current (our model)	Same	Current+Child
Intent	95.40	94.28	90.02
Slot	93.22	92.98	93.28

4.1. Syntactic Tying

An alternative to having a different matrix $W^{(t)}$ for each syntactic type as mentioned in Section 3.2.2, is to use a different matrix for each syntactic type of both the current node and the child node:

$$y = tanh(W^{(t,t_1)}x_1 + \cdots + W^{(t,t_k)}x_k + b^{(t)}) \quad (7)$$

where t is the syntactic type of the current node and t_i is the syntactic type of child i. Or, alternatively, to just use the same matrix for all syntactic types. It is not obvious which variant is best. However, experiments show that using just the syntactic type of the current node gives the best performance (Table 3). This suggests that the syntactic tree structure does help performance, and we expect that with additional training data, conditioning on the joint parent-child type would help further (there are over 300 different distinct pairs that occur, while there are only about 60 syntactic types).

4.2. Low-Rank Matrix Approximation

We found that using low rank matrices, coupled with high dimensional word vectors resulted in the best performance, both in terms of speed and accuracy (Table 4). In particular a rank of 1 along with a dimension of 300 (word vectors are dimension 100 due to usage of trigrams). The reason appears to be that a higher rank is much more prone to over-fitting, and a low-rank acts as a very good regularizer.

curacy numbers are averaged over 10-fold cross validation, so it was omitted from the comparison.

Table 4. Performance of different low-rank approximations

Rank	1 (our model)	3	10
Intent	95.40	93.83	93.49
Slot	93.22	93.21	92.46

Table 5. Joint learning

Model	Intent	Slot
RecNN intent only	95.17	–
RecNN slot only	–	93.48
RecNN joint (this work)	95.40	93.22

4.3. Joint Learning

Using one model to do both intent determination and slot filling simplifies the whole process. Instead of training and tuning two separate models, only one model needs to be trained. It also means that only one model is needed at testing time, again simplifying the system. The performance of the joint model is comparable to doing each task independently (Table 5), so there is no significant loss to not having two specialized models (and even a small gain for intent determination).

4.4. Cortana Data

We also evaluated our model on a real-world data set derived from Microsoft's personal assistant, Cortana. The Cortana data consists of user utterances from 3 domains: *calendar*, *communication* and *weather*, each having 10-20 intents, for a total of 55 intents. The utterances also have slot annotations, with a total of 83 slot labels (including the null label). There are 73653 training sentences and 8260 testing sentences. The data is roughly evenly split between the three domains. The baseline models only use trigrams as features.

Table 6 presents results for this data set. Overall, RecNN is close to the boosting baseline for intent classification. For domain classification, a boosting baseline got 98.65%, and RecNN achieved 98.23%. The domain classification result for RecNN was computed from the intent determination results on the 3 domains combined data set, which shows that even though some intents were incorrectly classified, the domain for those intents were more often correct. As for slot filling, the performance is a small improvement for 2 of the domains, as well as for the 3 domains combined data set.

Table 6. Cortana performance

Domain	Boosting	CRF	RecNN+Viterbi	
	Intent	Slot	Intent	Slot
Calendar	92.51	85.74	91.52	86.37
Communication	94.41	88.27	93.07	89.33
Weather	95.13	92.65	94.40	91.81
All 3	93.13	87.26	92.28	87.86

5. CONCLUSION AND FUTURE WORK

We have introduced recursive neural networks for joint semantic utterance classification and slot filling, and developed adaptations to specifically tackle these tasks (especially for slot filling). Performing intent determination and slot filling together in one model simplifies a dialog system, since only one model needs to be trained and tuned, and only one model needs to be used for prediction. Our model produces a performance level on par with other recently developed models, using purely lexical features and the parse tree. The syntactic tying experiments show that using the syntactic type of the nodes in the parse tree improves performance over ignoring that information, suggesting that exploring other methods of using that syntactic structure is worthwhile.

In future work we plan to incorporate explicit sequencing information into the model, by adding CRF dependencies between the slot labels. We also plan to explore the use of semantic parsers based on Abstract Meaning Representation [24], which may improve the correspondence between parse subtrees and slots.

6. REFERENCES

[1] Jordan B Pollack, "Recursive distributed representations," *Artificial Intelligence*, vol. 46, no. 1, pp. 77–105, 1990.

[2] Andreas Stolcke and Dekai Wu, *Tree matching with recursive distributed representations*, Citeseer, 1992.

[3] Christoph Goller and Andreas Kuchler, "Learning task–dependent distributed representations by backpropagation through structure," in *Neural Networks, 1996., IEEE International Conference on*. IEEE, 1996, vol. 1, pp. 347–352.

[4] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng, "Parsing with compositional vector grammars," in *In Proceedings of the ACL conference*, 2013.

[5] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 129–136.

[6] Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Advances in Neural Information Processing Systems*, 2011, pp. 801–809.

[7] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, 2013.

[8] Robert E Schapire and Yoram Singer, "Boostexter: A boosting-based system for text categorization," *Machine learning*, vol. 39, no. 2-3, pp. 135–168, 2000.

[9] Patrick Haffner, Gokhan Tur, and Jerry H Wright, "Optimizing svms for complex call classification," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*. IEEE, 2003, vol. 1, pp. I–632.

[10] John Lafferty, Andrew McCallum, and Fernando CN Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001, pp. 282–289.

[11] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," *INTERSPEECH*, pp. 1605–1608, 2007.

[12] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *INTERSPEECH*, 2013.

[13] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, "Recurrent conditional random fields for language understanding," in *ICASSP*, 2014.

[14] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for language understanding," in *INTERSPEECH*, 2013.

[15] Minwoo Jeong and G Geunbae Lee, "Triangular-chain conditional random fields," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 7, pp. 1287–1302, 2008.

[16] François Mairesse, Milica Gasic, Filip Jurcícek, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young, "Spoken language understanding from unaligned data using discriminative classification models," in *ICASSP 2009*. IEEE, 2009, pp. 4749–4752.

[17] Puyang Xu and R. Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, Dec 2013, pp. 78–83.

[18] Richard Socher, Q Le, C Manning, and A Ng, "Grounded compositional semantics for finding and describing images with sentences," in *NIPS Deep Learning Workshop*, 2013.

[19] Karl Moritz Hermann and Phil Blunsom, "The role of syntax in vector space models of compositional semantics," in *ACL*, Sofia, Bulgaria, August 2013.

[20] Geoffrey Hinton, Nitsh Srivastava, and Kevin Swersky, "Neural networks for machine learning," `www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`, Accessed 2014-05-31.

[21] G. Tur, D. Hakkani-Tr, and L. Heck, "What is left to be understood in atis," in *IEEE SLT Workshop*, 2010.

[22] G. Tur, D. Hakkani-Tur, L. Heck, and S. Parthasarathy, "Sentence simplification for spoken language understanding," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, May 2011, pp. 5628–5631.

[23] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, "Use of kernal deep convex networks and end-to-end learning for spoken language understanding," in *SLT*. IEEE, 2013, pp. 210–215.

[24] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider, "Abstract meaning representation for sembanking," 2013.