Proceedings of 9th IEEE International Conference
on CYBER Technology in Automation, Control, and Intelligent Systems
July 29- August 2, 2019, Suzhou, China

TLDR: Syntaxic tree and task/dependencies extraction

Natural Spoken Instructions Understanding for Robot with Dependency Parsing

Yadeng Chai and Yong Liu*, *Senior Member, IEEE*

***Abstract*—This paper presents a method based on syntactic information, which can be used for intent determination and slot filling tasks in a spoken language understanding system including the spoken instructions understanding module for robot. Some studies in recent years attempt to solve the problem of spoken language understanding via syntactic information. This research is a further extension of these approaches which is based on dependency parsing. In this model, the input for neural network are vectors generated by a dependency parsing tree, which we called window vector. This vector contains dependency features that improves performance of the syntactic-based model. The model has been evaluated on the benchmark ATIS task, and the results show that it outperforms many other syntactic-based approaches, especially in terms of slot filling, it has a performance level on par with some state of the art deep learning algorithms in recent years. Also, the model has been evaluated on FBM3, a dataset of the RoCKIn@Home competition. The overall rate of correctly understanding the instructions for robot is quite good but still not acceptable in practical use, which is caused by the small scale of FBM3.**

I. Introduction

The goal of spoken language understanding is to transform textual information into semantic representations that can be processed by machines. Semantic representations are usually described as intents and slots. In order to understand natural spoken instructions, a robot is expected to be able to recognize the intent and slots in an instruction. For instance, a robot user might ask the robot to do something by saying that "Hey robot, carry this mug to the desk", the robot would get the intent: "bringing", and a list of slots like: theme: "this mug", goal: "to the desk". These above are the two major tasks in SLU which also needs to be solved in the problem of natural spoken instructions understanding for robot: intent determination (ID) and slot filling (SF).

The research about SLU emerged in the 1990s when DARPA started the ATIS project [1], since then the ATIS dataset becomes a benchmark to evaluate the performance of a SLU system. For intent determination, researchers found Adaboost [2] and SVM [3] performs well. On the other hand, in the field of natural language processing, the slot filling task is usually regarded as a sequence labeling problem. Common solutions are based on the HMM model, the maximum entropy model, and the CRF model [4], which is the mainstream method to solve the problem of sequence labeling. Nowadays with the development of deep learning, RNN shows its strong ability in the problem of sequence labeling, and the end-to-end in deep learning also makes it easier. Although SLU has been studied for many years, it is still not possible to say that it is a solved problem because of the variety of spoken language and the ambiguity of word sense. Some syntactic-based approaches have emerged before the days when neural networks and word embedding are widely used in SLU. Early SLU systems such as SRI Gemini, MIT TINA, and CMU Phoenix systems [5] rely heavily on syntactic parsing. Jeong and Lee used trigger patterns which performed well on the Communicator dataset. They also tried to make use of syntactic information such as head word but failed. Moschitti et al. presented the first study using syntactic features for slot filling via syntactic tree kernels with support vector machines. This improved the accuracy performance on ATIS to over 95%. Gokhan Tur et al. presented an approach populating heterogeneous features from syntactic and semantic graphs of utterances, which achieved good results on the classifier based on word n-grams. Inspired by keyword search, Gokhan Tur et al. also proposed a sentence simplification model [6], which uses the keywords of a sentence to classify the intents and slots. The sentence simplification model achieves good results in intent determination task, which is mainly because of that the intention expressed by the user is usually determined by one or two keywords in the sentence especially the verbs. For instance, when words like "take" or "get" appears in a sentence, it is easily to conclude that the user's intention is "bring", which means to command the robot to bring something back.

In recent years, deep learning models like RNN and LSTM has been widely used in natural language processing. LSTM networks [7] can capture long-distance dependencies that would make better use of historical information of a sentence, thus has been widely used in natural language parsing[8]. Some researchers have tried to capture syntactic information of a sentence via neural networks. Guo et al. proposed a RecNN model with syntactic tree [9]. The input of this model is each single word vector, and each part of speech is regarded as a weight vector, thus, the embedding of each word in its path is a dot product of the word vector and the part-of-speech weight vector. If a parent node has multiple child branches, the embedding would be generated as the sum of each branch. After such a dot product operation, a vector is finally output at the root node of the parse tree. This vector is generated from the vector of all words in the sentence which represents the entire sentence, so it can be used to classify the intent. On the other hand, path vector is used for slot filling task, for an

This work was supported in part by China National Science Foundation under grants 61473155, by Jiangsu Technology Department under Modern Agriculture BE2017301, by Equipment Prereserch and Sharing Technology Project 41412040102, by Jiangsu Qing Lan Project, by Six talent peaks project in Jiangsu Province GDZB-039, and by Nanjing Science & Technology Plan Project 201608026.

Yadeng Chai is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, 210094 China. (E-mail: 2863511499@qq.com)

*Corresponding Author: Yong Liu is with School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (E-mail: liuy1602@njust.edu.cn).

978-1-7281-0770-7/19/$31.00 ©2019 IEEE

example, the path of word "in" in the syntactic tree is "IN-PP-NP", so its path vector is a vector weighted by the path "IN-PP-NP". In the end, a concatenation of the path feature of three words is regarded as the tri-path feature to classify the slot, and finally the sequence labeling task at sentence level is optimized by Viterbi algorithm.

Although Guo's RecNN model uses deep learning methods, the results are not as good as sentence simplification. The accuracy and F1-score on ATIS are also 1% - 2 % lower than many other deep-learning-based SLU models in recent years. Zhang et al. analyzed this, and they thought that this may be because the scale of the dataset is too small such that human-written syntactical features perform better.

As seen, while using syntactic information for SLU is not a novel idea, now we take the RecNN model as a reference and propose going one step further to make syntactic information better integrated with deep learning. The following two reasons were considered: First, in the RecNN model, for the same sentence, a slight change may result in a completely different syntactic tree in syntactic parsing, for example, "Please show me the flights from New York to London" and "Can you show me the flights from New York to London", the syntactic tree of the above two will be very different, which will greatly affect the results of classification, whether for intent determination or slot filling. In order to solve this problem, a dependency parsing [10] tree is tested instead of syntactic tree in our model, although the change of a sentence will also lead the structure of the dependency parsing tree to change, the dependency relationship between each two words usually remains the same. For example, in the above two sentences, "London" will always depend on "flights", while "flights" always depends on "show". Dependency parsing indicates the association between words, usually this kind of association only relates to the semantics of the sentence, so classifiers based on dependency parsing works better. Second, the input single word vector is obtained through neural networks training, whether with word2vec method or GloVe method [11], after large-scale text training, a word is converted into a dense vector, for similar words, their word vectors are similar, otherwise semantically independent of each other. A classic example would be v("King") − v("Man") + v("Woman") = v("Queen").But, if the word vector is weighted and summed for several times, the semantics of the vector representation will be changed. That is why the operation between the word vectors needs to be simplified by concatenating the vectors of words with dependencies and input them directly into the classifier. The next section presents the process of building word embedding with dependency parsing and how to use it for intent determination and slot filling. The experimental results are presented in Section 3, and the conclusions are given in Section 4.

II. MODEL

A. Spoken Language Understanding Model

For a user's utterance S, the SLU system is expected to predict the user's intent $\hat{c}_i$ and slot label sequence $\hat{l}$. Suppose that there are N kinds of intents:

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

TABLE I. IOB ANNOTATION

Sentence	put	the	pillow	under	the	bed
Slot Label	O	B-theme	I-theme	B-goal	I-goal	I-goal
Intent	PLACING					

For the slot filling task, each word in the sentence is annotated in IOB format. TABLE I shows an example of an annotation: Where B- represents the beginning of a label, I- represents the non-starting part of a label, and O represents that the word does not belong to any label. The SLU system is expected to predict that the word belongs to B- or I- or O:

$$\hat{l} = \underset{l \in L(s)}{\arg\max}\, p(l \mid s) \tag{2}$$

B. Word Embedding with Dependency Parsing Tree

The dependency parsing theory was proposed by the French linguist L. Tesniere in 1959, which has a great significance to the development of linguistics. According to the axioms of dependency parsing, in any kind of sentence, there is only one component which is independent, while other components are directly dependent on it or its child. In addition, any component cannot depend on more than one components. Figure 1 shows an example of dependency. A sentence can be parsed to generate a dependency parsing tree, and the core verb in the sentence is the central component that governs other components, but for the central component itself, it is not subordinate to any other components. All the dominant components are dependent on each other with a kind of dependency relationship.

In a dependency parsing tree, "dependency" refers to the relationship of dominance between each two words, and this kind of relationship is asymmetric and directional. The dominant word is usually called a governor or regent, while the dominated word is usually called a modifier or subordinate. The dependency parsing structure has many different types, in fact, it is a list of modification or collocation relationships between each two words in a sentence, which is connected by a governor and a modifier. For example, in the sentence "there is some bread on the desk", "some" depends on "bread", and the relationship between them is "det", which refers to a determinermodifier relationship. In this sentence, "is" is the core verb, so it is in the root node of the dependency parsing tree, while other nodes are derived from it.

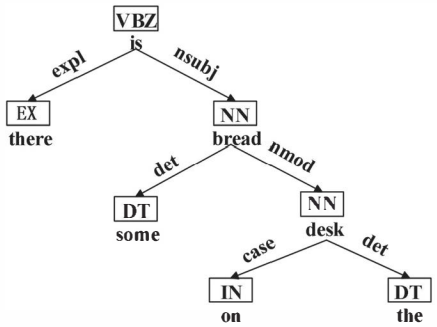


Figure 1. A dependency parsing tree

In this research, Stanford coreNLP[12] is used as a parsing tool to generate the dependency parsing tree for a sentence. Stanford coreNLP is an annotation-based natural language processing system that provides a variety of commonly used natural language processing functions, including word segmentation, part-of-speech tagging, named entity recognition, etc. To simplify the problem, the dependency between words like "det" or "nmod" or something else is not focused on.The only one considered is the order of dependencies between each two words.

Like many other researchers, our idea is to incorporate syntactic information into word embeddings, after that these processed word embeddings will be used to train a neural network, so that the deep learning model can gain the ability to classify labels with syntactic information. The way of processing word vectors is as follows: For a sentence $S=\{w_1,w_2,w_3...w_k\}$, for each word w_i in the sentence, suppose that its parent node in the dependency parsing tree is w_i^1, and the parent node of w_i^1 is w_i^2, by analogy, a dependency sequence $D(w_i)$ of the word w_i can be listed as: $D(w_i)=\{w_i^1, w_i^2...w^{root}\}$, here w^{root} is the root node of the dependency parsing tree.

Since the distance between each word and the root node is different, the length of $D(w_i)$ for each word would be also different. In order to make the embedding length of each word the same, a padding of each sequence $D(w_i)$ is built to generate a text window, which we called dependency window. The size of the window is set to d, where $2 \leqq d \leqq 6$, because in all sentences of our dataset, the maximum depth of the dependency parsing tree is 6. If the length of a word's $D(w_i)$ is less than d, w_i^1, w_i^2...will be added to the end of the sequence until it reaches d. On the contrary, if $D(w_i)$ is longer than d, its first d elements will be put into use. For example, in the sentence "there is some bread on the desk", if the size of the dependency window is 6, the dependency window of "on" can be represented as {on, desk, bread, is, on, desk}.

After generating the dependency window of w_i, each w_i^j in the dependency window will be mapped to a 50-dimensional vector via the GloVe pre-trained word vectors. Then the concatenation of these w_i^j is regarded as the embedding of word w_i:

$$e(w_i)=[e(w_i^1),e(w_i^2),e(w_i^3)...] \tag{3}$$

Every word in a sentence will be finally mapped to a dimension of $50\times d$ vector, and the final input vector of the neural network is the concatenation of $D(w_i)$, which we called a window vector:

$$e(S)=[e(w_1),e(w_2),e(w_3),e(w_4)...] \tag{4}$$

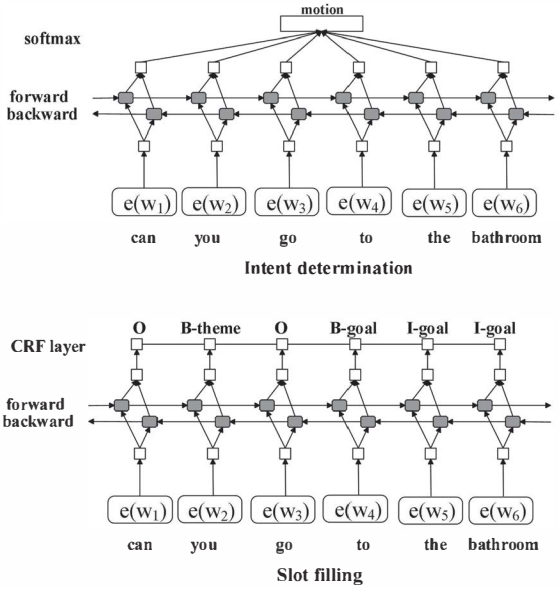


Figure 2. The structure of our model

C. *Intent Determination and Slot Filling*

A bidirectional LSTM network is used as the encoder in this approach, and the input of each time step for bidirectional LSTM is $e(S)$ above. The concatenation of the output hidden state sequence of the forward LSTM and backward LSTM is regarded as the complete hidden state sequence:

$$h_t=[\overrightarrow{h_t},\overleftarrow{h_t}] \tag{5}$$

For intent determination, a softmax layer is accessed behind the BiLSTM layer which gives the probability of each label. For slot filling, a CRF layer behind the softmax layer generates the final output. The CRF layer has an ability to make use of transfer feature by checking the order between each two output labels, this would prevent problems like a "B-theme" after "B-theme" and finish the sequence labeling task on a sentence-level. The entire model is shown in Figure 2.

III. Experiment and Results

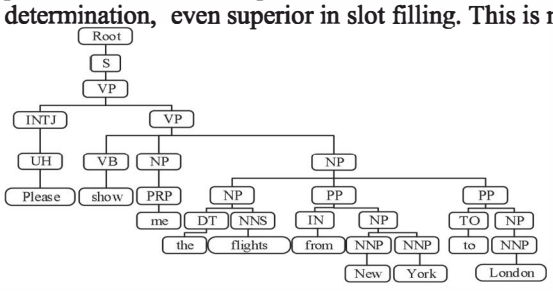
A. *Results on ATIS*

In order to compare with other machine learning models, the approach in this paper is performed on the benchmark ATIS dataset. The ATIS dataset was collected by DARPA in the early 1990s. The ATIS dataset includes verbal queries about flight related information like "I want to go from Boston to Atlanta on Monday". All the tags have been encoded in IOB format. The ATIS training set and test set includes 4,978 and 893 sentences, and the average sentence length is 15. The number of different slots is 128, including O label (NULL). In addition, each number in the sentence is replaced with string "DIGIT".

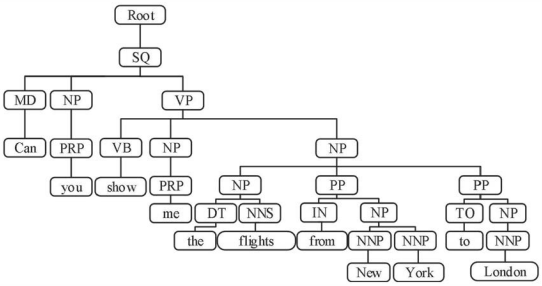
TABLE II. COMPARISON BETWEEN RECNN AND OUR MODEL

Model	Intent(accuracy)	Slot(F1)
RecNN	95.40	93.22
Our model	95.43	95.32

The results compared with RecNN are shown in TABLE II. As seen, the SLU model based on dependency parsing has a performance level on par with RecNN in terms of intent determination, even superior in slot filling. This is mainly due to the word vector operation. As mentioned in section 1, a syntactic tree can be easily changed by sentence transformations, for an example, "Please show me the flights from New York to London" and "Can you show me the flights from New York to London", the syntactic tree is shown in Figure 3, and the dependency parsing tree is shown in Figure 4.

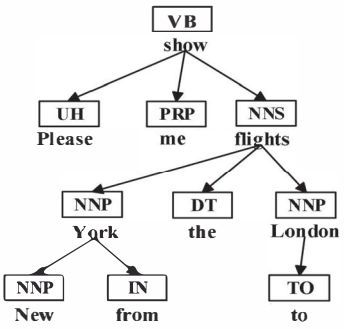


(a) syntactic tree of sentence "Please show me the flights from New York to London"

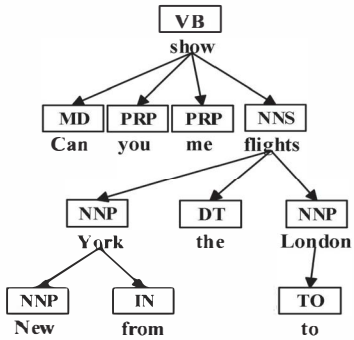


(b) syntactic tree of sentence "Can you show me the flights from New York to London"

Figure 3. Syntactic tree of two sentences



(a) dependency parsing tree of sentence "Please show me the flights from New York to London"



(b) dependency parsing tree of sentence "Can you show me the flights from New York to London"

Figure 4. Dependency parsing tree of two sentences

In these two sentences, both of the slot of "London" are the same. However, according to the method of RecNN, the path vectors of "London" in these two sentences are different because their paths in the syntactic tree are very different, this can be seen between graph (a) and graph (b) in Figure 3. But in our model, each word is mapped to a vector generated from words with dependency relationship. In the above example, "London" always depends on "flights", while "flights" always depends on "show", therefore, the embedding of "London" in the two sentences are exactly the same, this can be seen between graph (a) and graph (b) in Figure 4. The above example shows that, on a slot filling task, dependency parsing can eliminate the influence of slight changes in a sentence, although this does not always occur in every situation. Sometimes even if a sentence changes a lot, the dependencies between each two words still exists, this makes our model more robust in slot filling.

B. Performance of operation optimization

Another reason why slot filling is improved can be explained from vector operations. To evaluate the performance of our algorithm on the problem of spoken instructions understanding for robot, FBM3, a dataset from Robot Competitions Kick Innovation in Cognitive Systems and Robotics (RoCKIn) was chosen. FBM3 collects some spoken commands such as "PLACING", "TAKING", etc., which have been translated into text. The intents and slots in the instructions have been labeled by hand. The training and test sets contain 306 data and 100 data, which has a total of 388 different words. For the propose of better training, some data in the original dataset were filtered out and finally 12 different kinds of intents and 10 different kinds of slots were collected. These data containing different intents and slots are evenly divided into a training set and a test set according to a ratio.

In our dependency model, the concatenation of each single word vector is the input vector for neural network, rather than a weighted summation. The goal of capturing syntactic information is to change the pre-trained word vector as little as possible. Word vectors are also known as distributed representations of words, which are usually dense, low-dimensional, and of practical value. Word vectors have been proven empirically to follow linguistic laws. For example, researchers have found that the gap between word vectors of similar words has a clustering center, which indicates that the semantics of words can be expressed by word vectors. When considering the usage of syntactic information to solve the problem of spoken language understanding, a usual method will be to weight and sum the words according to their path in the syntactic tree, so that the syntactic information will be encoded into the vector. The problem with this will be that the words that are originally semantically related can be shifted in the high-dimensional space because of the operation of the word vector, and this kind of change is usually uncertain. This will cause that the semantics of words be changed, also the classification criteria of the classifier will be changed.

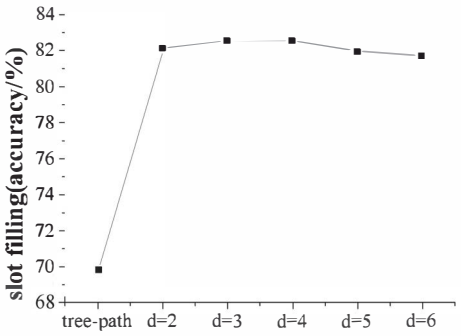


Figure 5. Comparison between path vector and concatenation of vector

To prove this, a comparison experiment on FBM3 has been performed. The words were weighted according to their path in the syntactic tree, then compared with the method based on dependency window. The size of the dependency window was changed several times to ensure that the experiment is believable. Although FBM3 is a small-scale dataset, the effect of the operations between word vectors on the classification results can still be seen in Figure 5.

It can be seen that the result of the weighted operation is much worse than the method of concat, although the size of dependency window also has an effect on the accuracy rate, it can be omitted from the comparison with the effect of the vector operation. While making use of syntactic information of a sentence, the RecNN model changes the semantics in the word vector. Now this is replaced with a simpler concat operation, which is the other reason why our model has an improved performance in slot filling.

As seen, syntactic information can be well used through our improved method. Dependency parsing may not be the best way to make use of syntactic information, but its contribution to spoken language understanding shows the enormous potential of syntactic information, particularly in deep learning applications, we have not seen such a good instance.

C. Comparison

Comparing the overall performance of the algorithm on FBM3 and ATIS, the results are shown in TABLE III. As seen, our model shows good generalization ability on both FBM3 and ATIS, which indicates that dependency model is robust enough. Both intent determination and slot filling results on FBM3 are worse than ATIS, this is because their data scale is very different from each other, and the average length of the sentence in ATIS is longer than FBM3, which may lead BiLSTM to capture longer dependency information on ATIS. The scale of data has a great effect on the results. Some researchers use the entire FBM3 dataset as a training set and evaluate it on another dataset. Their intent determination result is better than ours. However, our slot filling results are better than theirs. This is mainly due to the effect of the CRF layer which optimizes the order of the labels in the sequence.

TABLE III. COMPARISON BETWEEN FBM3 AND ATIS

	Intent (accuracy)	Slot (accuracy)	Slot (F1-score)
FBM3	81.00%	81.57%	65.37%
ATIS	95.43%	98.14%	95.32%

TABLE IV. COMPARISON WITH PREVIOUS APPROACHES

	intent (accuracy /%)	slot (F1-score /%)
CRF	–	92.94
RNN	–	95.06
Boosting	95.50	–
Sentence simplification	96.98	95.00
RecNN	95.40	93.22
RecNN+Viterbi	95.40	93.96
BiGRU+CRF	98.10	95.49
Our model	95.43	95.32

In addition, on FBM3, the accuracy of slot filling differs greatly from F1-score, which may be caused by the uneven distribution of the slot labels on FBM3. The results of our model on FBM3 is quite good but still not applicable in practical situation. This is mainly due to the small scale of data. To improve this, a larger dataset is needed.

Comparing with other algorithms that were trained and evaluated on the ATIS benchmark dataset, the performance of our algorithm can be measured. The results are shown in TABLE IV. Also, a comparison between the use of RNN and LSTM was given, which does not have a CRF layer. The results are shown in TABLE V, as expected, the accuracy achieved when using LSTM was higher. This can be caused by the existence of long-term dependencies that could not be memorized by RNN, because some of the commands in FBM3 are as extensive as 15-19 words.

Since the ATIS dataset have been studied for more than ten years and the score of the state-of-the-art method is very high, the absolute improvement may be not very high. It can be seen that our model has the same recognition ability as several other models in recent years, especially in the case of slot filling, our model even surpasses some traditional models. This indicates that the dependency parsing information can be well used for SLU, and the spoken language understanding model based on syntactic information has good generalization performance. Sentence simplification also performs very well, especially in the case of intent determination, however, it relies on a number of hand-written syntactic parse tree transformations, which would not easily adapt to varied spoken language understanding tasks, including the robotic instructions understanding task.

IV. CONCLUSION

This paper presents a method that captures the dependency information of sentences via syntactic parsing tools. A kind of neural network was trained with these information. The results show that this model produces a performance level on par with several other state of the art SLU models. The results on ATIS indicate that syntactic information can be well used for spoken

TABLE V. COMPARISON WITH RNN AND LSTM

	RNN	LSTM	Our model
Intent(accuracy)	74.00%	80.00%	81.00%
Slot(F1-score)	44.91%	55.32%	65.37%

language understanding. Experiments on FBM3 show that the accuracy and F1-score in intent determination and slot filling tasks reach 81.00% and 65.37%. This result is quite good, but still not applicable in practical applications, which is mainly limited by the scale of training dataset.

In the future work, we plan to retrain the word vectors via dependency parsing, so that the word embeddings will be better integrated with the dependency model than the pre-trained GloVe word vectors, thus improve the performance of the model. In addition, in order to build a more robust robotic instruction understanding model, we need to find a larger dataset or collect more data by hand.

REFERENCES

[1] Hemphill, Charles T., John J. Godfrey, and George R. Doddington. "The ATIS spoken language systems pilot corpus." *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*. 1990.

[2] Schapire, Robert E., and Yoram Singer. "BoosTexter: A boosting-based system for text categorization." *Machine learning* 39.2-3 (2000): 135-168.

[3] Haffner, P. , G. Tur , and J. H. Wright . "Optimizing SVMs for complex call classification." *IEEE International Conference on Acoustics* IEEE, 2003.

[4] Raymond, Christian, and Giuseppe Riccardi. "Generative and discriminative algorithms for spoken language understanding." *Eighth Annual Conference of the International Speech Communication Association*. 2007.

[5] Ward, W. . "Understanding spontaneous speech: the Phoenix system." *International Conference on Acoustics* IEEE, 1991.

[6] Tur, Gokhan, et al. "Sentence simplification for spoken language understanding." U.S. Patent Application No. 15/271,859. 2017.

[7] Hochreiter, S, and J. Schmidhuber. "Long Short-Term Memory." *Neural Computation* 9.8(1997):1735-1780.

[8] De Lhoneux, Miryam , M. Ballesteros , and J. Nivre . "Recursive Subtree Composition in LSTM-Based Dependency Parsing." (2019).

[9] Guo, Daniel, et al. "Joint semantic utterance classification and slot filling with recursive neural networks." *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014.

[10] Vardi, Moshe Y. *Fundamentals of dependency theory*. IBM Thomas J. Watson Research Division, 1985.

[11] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.

[12] Manning, Christopher, et al. "The Stanford CoreNLP natural language processing toolkit." *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014.