RESEARCH ARTICLE OPEN ACCESS

# A Study of NLP Methods for Converting Natural Language into Programming Language

Alind D. Naik [1], Prof Dr. J. A. Laxminarayana [2]
Department of Computer Science and Engineering
, Goa College of Engineering
Goa University - India

**ABSTRACT**
The implementation of automated solution requires time and some knowledge about the system and its working. Many researchers have proposed the automated solutions to save time but they require the input from user. The interaction with different applications may be a tedious task for a user as different applications will require different user inputs. The simplest way for these automations is to support natural language input. The main problem is to choose optimal language processing model. Without an optimal language processing model, there can be no learning and processing the natural language input from the user. We explore SyntaxNet, which is an approach for natural language processing, that can be used for converting a natural language into a programing language.

***Keywords :—*** NLP, natural language processing model, Parsey McparseFace, NLP to Programing language, dependency parser, Treebank, Transition base dependency parser.

## I. INTRODUCTION

Natural language processing (NLP) is a domain which is concerned with understanding and processing natural languages. With NLP we can use natural language as an interface between human and machines. Natural language such as English is a very complex language and it is difficult for natural language processing algorithms to process it. Now there is ample amount of data available to build a data driven model, so that we get a more precise NLP model.

For understanding linguistics and natural language processing, dependency based grammar has played an important role. Dependency based grammar is highly motivated by the efficiency that results from more constrained parsing problem for these type of representation and the usefulness of bi-lexical relations in ambiguity problems in a language [1].

In this paper we discuss the use of SyntaxNet based model which is a data driven model that is trained on large datasets. These datasets are also known as Treebank. This model uses transition based dependency parsing to parse a natural language sentence and produce a dependency parse tree which gives the semantic meaning and the structure of a sentence. This dependency tree can be used in various applications of natural language processing [2]. In this paper we show how SyntaxNet models can be used to parse natural language input and how the output of this model can be processed.

## II. TRANSITION BASED DEPENDENCY PARSER

Transition based dependency parser carries out a word by word linear scan over the input sentence to build a dependency parse tree [12]. At every iteration parser maintains a partial parse tree created by examining the part of sentence. For maintaining this partial parse tree parser uses a stack to store the words which are currently being processed. Parser also keeps a buffer to store the words of the sentence that are not processed. The parser than carry on to apply transitions to the words remaining in the buffer till all the words in buffer are processed and buffer get empty. The output of dependency parser after processing entire sentence is a complete dependency parse tree.

The initial step of the parsing process is to have the sentence tokenised and loaded on the buffer. The initial step creates an empty ROOT node on the stack. The parser uses three transitions that are applied to each word processed. The left arrow marks the second word on the stack is a dependent of the top word on the stack, and then removes the second word from the stack if the stack contains at least two words. The right arrow mark the top word on the stack is a dependent of the second word on the stack, and pops the first word from the stack if the stack contains at least two words and the shift transition removes a word from the buffer and pushes it onto the stack provided that buffer is not empty.

With these three types of transitions, a parser can generate dependency parse tree for any given English sentence. With each transition parser also specifies the type of the relationship between the head and dependent being described. The parser decides among transitions at each state using a neural network classifier which uses trained dataset (Treebank III) containing dependency parse tree for every sentence it can encounter. The parser current parse tree of already parsed words are provided as input along with the new word to the classifier, so parser then chooses among the possible transitions to take for the new word. These representations describe various features of the current stack and buffer contents in the parser state [2].

Example of dependency parse tree for the input “Alice saw Bob in the alley.” can be seen below.

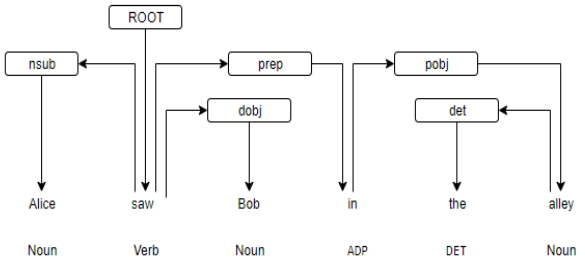


Fig. 2.1 Dependency parse tree

## III. TREEBANK

Human beings around the globe have drawn in the syntactic structures of sentences of various languages as dependency graphs and these are referred to as Treebank. People spend years working on creating CFG (Context free grammar) for specific language but no one really uses them whereas Treebank is a valuable resource of linguistic. Examples of Treebank are Penn Treebank which contains standard POS tagging[3], the English "Treebank Union" multi-domain corpus containing data from the OntoNotes corpus[4] and the English Web Treebank [5].

## IV. SYNTAXNET

SyntaxNet [7] is a framework for transition based dependency parser, it is the main component of many system that uses natural language processing. It takes a sentence as an input, it than tags each word in that sentence with a part-of-speech (POS) tag that identifies the word's syntactic function, it also identifies the syntactic relationships between words. These relations are represented in the form of dependency parse tree. This syntactic relationship between the words and the position of these words in the dependency parse tree provides enough information to understand the meaning of the input sentence.

It is an open source framework which is available online[1]. SyntaxNet can be downloaded and the model can be trained with application specific Treebank dataset. SyntaxNet uses universal dependency. The most complex trained SyntaxNet models are also available as open source API.

In English language sentences can have multiple meanings giving raise to ambiguity problem in language processing model. To resolve the ambiguity problem in a language SyntaxNet uses neural networks to take best decision. It does so by processing the input sentence from left to right using stack and buffer as discussed in section II. The dependencies between words are added one after the other as each word in the sentence is processed incrementally. At particular point in processing, parser uses neural network scores for making plausible decision to resolve ambiguity problem. For this reason, it is uses beam search in the model. Instead of taking the first best decision at every point, it keeps multiple hypotheses for the processed sentence and chooses the best upon processing the entire sentence. Hypotheses that have higher rank are kept under consideration while remaining hypotheses are discarded [7].

## V. PARSEY MCPARSEFACE

Prasey McParseface[3] is a model based on SyntaxNet which is trained with the TensorFlow[2] framework. Parsey Mcparseface is trained on English corpus. This model accepts English sentence as input and give dependency parse tree along with part-of-speech tags as discussed in section II. [7] have presented a simple and yet powerful model architecture that Parsey McParseface uses, that produces best results for POS tagging and dependency parsing. This model combines the transition-based parsing algorithms and the modeling power of neural networks. [7] demonstrated that feed-forward network[8] without recurrence can outperform recurrent models such as LSTMs(Long short-term memory) when they are trained with global normalization transition based processing while being significantly faster. Global normalization helps the model overcome the label bias problem from which locally normalized models suffer.

This model performed dependency parsing on the Wall Street Journal and achieved the best ever published unlabeled attachment score of 94.61%. Parsey McParseface is pre-trained, state-of-the art English dependency parser, which is tuned for a balance of speed, simplicity, and accuracy [7].

Parsey McParseFace model is available for developers on Google's Deep AI platform for free with limited access1. Developers can directly copy the code and use the API key for the access of this model. Paid version is also available with no restriction on usage. It takes English sentence as input and generates dependency parse tree. This dependency parse tree contains the part-of-speech tag and bilingual dependency labels. The output of the model is represented in the form of JSON data structure [9]. This JSON data can be parsed to extract the required information from the parse tree. The example of parse tree generated by the parser on the input of sentence "Alice saw Bob in the alley." is shown in fig. 5.1.

Output of the Parsey McParseface is shown in above figure. It is in Json(JavaScript Object Notation) that can be parsed to extract the labels and dependency between the lables. The root node is represented as ROOT which contain three data fields index, token, tree, pos and label.

1. Index –the position of the token in sentence.
2. Token –the word from the sentence it referring to.
3. Tree – it is the sub-tree of the current node in dependency tree.
4. POS – Part-of-speech tag
5. Label – gives the label for the word( like noun, verb, determiner etc.)

```
[
  {
    "tree": {
      "ROOT": [
        {
          "index": 2,
          "token": "saw",
          "tree": {
            "punct": [
              {
                "index": 7,
                "token": ".",
                "pos": ".",
                "label": "."
              }
            ],
            "dobj": [
              {
                "index": 3,
                "token": "Bob",
                "pos": "NNP",
                "label": "NOUN"
              }
            ],
            "prep": [
              {
                "index": 4,
                "token": "in",
                "tree": {
                  "pobj": [
                    {
                      "index": 6,
                      "token": "alley",
                      "tree": {
                        "det": [
                          {
                            "index": 5,
                            "token": "the",
                            "pos": "DT",
                            "label": "DET"
                          }
                        ]
                      },
                      "pos": "NN",
                      "label": "NOUN"
                    }
                  ]
                },
                "pos": "IN",
                "label": "ADP"
              }
            ],
            "nsubj": [
              {
                "index": 1,
                "token": "Alice",
                "pos": "NNP",
                "label": "NOUN"
              }
            ]
          },
          "pos": "VBD",
          "label": "VERB"
        }
      ]
    },
    "sentence": "Alice saw Bob in the alley."
  }
]
```


Fig. 5.1 Output of Parsey McParseFace

The dependency between the children are given by the labels such as “ROOT”, “punct”, “dobj”, “prep” and “nsub” in fig. 5.1. These dependencies between the words are described in dependency manual [10]. This dependency manual contains approximately 50 grammatical relationships between the words in English language. These dependencies are binary relationship between the governor (parent node) and the dependent (child node). In figure 5.1 the “saw” is the parent node and the dependency label between “saw”(parent node) and “Bob”(child node) is “dobj”. Similarly dependency label for words “saw”(parent node) and “Alice”(child node) is “nsub”. Where “dobj” is a directed object (saw, Bob), “nsub” is a nominal subject (Alice, saw) , “prep” is a prepositional modifier (saw, in), “pobj” is an object of preposition (in, alley),and “saw” is the root[10].

### *A. Links and Bookmarks*

1 github.com/tensorflow/models/tree/master/syntaxnet
2 www.tensorflow.org
3 deepai.org/ai-text-processing

## VI. CONCLUSIONS

Parsey McParseFace is a model base SyntaxNet which is an open source tool which is freely available. Parsey McParseFace is a state-of-the art pre-trained model for English dependency parsing. No additional training is needed for this model and can be directly used to get dependency parse tree for English Language. For other languages developers can download the SyntaxNet framework and can train the model with language specific Treebank. Since Parsey is pre-trained it is simple to use and understand the linguistic structure of English grammar. Output of these can be directly used as an intermediate step while converting natural language into programing language. SyntaxNet can be used for sentiment analysis, text summarization and text tagging..

## REFERENCES

[1] Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In Proceedings of

the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

[2] Mrd. Ing. Alexandru Trifan, Mrd. Ing. Marilena Angheluș, Ș.L. Dr. Ing. Rodica Constantinescu. Natural Language Processing Model Compiling Natural Language into Byte Code Informatics Engineering and Computer Science, Faculty of Electronics, Telecommunications and Technology of Information.

[3] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics.

[4] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In Proceedings of the Human Language Technology Conference of the NAACL, Short Papers.

[5] Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

[6] Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing.

[7] David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov and Michael Collins, 2017. Globally Normalized Transition-Based Neural Networks Daniel Andor, Chris Alberti, Google Inc New York, NY.

[8] David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics.

[9] Felipe Pezoa, Juan L. Reutter, Fernando Suarez, Martín Ugarte, Domagoj Vrgoč, 2016. Foundations of JSON Schema. Proceedings of the 25th International Conference on World Wide Web Pages 263-273

[10]. Marie-Catherine de Marneffe and Christopher D. Manning, 2008. Stanford typed dependencies manual, COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation.