Dependency Parsing

Gurvinder Singh
University at Buffalo
gurvind2@buffalo.edu

Harshit Monish
University at Buffalo
hmonish@buffalo.edu

Abstract

In this project, we plan to implement the transition based and graph based dependency parser using neural network approaches for feature representation to extract the syntactic interpretations. We will further explore different approaches to generate the word embedding using Random Embedding, Glove embedding, Elmo embedding CSKG embedding. Our main contribution will be applying graph based embedding in transition based dependency parsing and graph based dependency parsing approach. Finally we will perform a comparative analysis of these approaches using the Universal Dependencies dataset [2].

1 Introduction

Dependency parsing is one of the most fundamental tasks in the field of Natural Language Processing and is used mostly to facilitate further NLP research and applications. Dependency syntax postulates that syntactic structure consists of relations between lexical items, normal binary asymmetric relations called dependencies. Building accurate dependency parsing models is vital as a small error or mistakes made by the parser results in big errors into downstream tasks. The dependency parser development is dependent on treebank annotation of grammar (part of speech, morphological features and syntactic dependencies) which are available readily, one such framework is Universal Dependencies framework consisting of cross-linguistically consistent treebank annotations for many languages for multilingual parser development.

Transition based dependency parsing architecture draws on shift-reduce parsing. We have a stack on which we build the parser, a buffer of tokens to be parsed and a parser which takes actions on the parse via a predictor.

Graph based dependency parsing architecture is represented as a directed graphs G = (V, A), consisting of a set of vertices V, and a set of ordered set of pairs of vertices A, called arcs. The set of vertices correspond to the set of words, punctuation in a given sentence and the set of arcs captures the head-dependent and grammatical function relationships between the elements in V. Most common parsing approach is to create a rooted dependency tree as a directed graph having single designated root node that has no incoming arcs and each vertex has exactly one incoming arc with a unique path from the root to each vertex V.

In this project, we propose to build a transition based and a graph based model for dependency parsing. We will first explore different word embedding generation mechanisms in each model and then apply a Bi-LSTM model in transition based model to predict the transition step (Shift, Left-Arc, Right-Arc, Reduce) and similary a Bi-LSTM model in graph based model to find the maximum probability label for each edge in a particular parse, similar to Dozat and Manning 2017, Dozat, Qi and Manning 2017 [4] [5] and Kipperwasser and Goldber 2016 [7] approach.

2 Related Work

Modern approaches to dependency parsing can be broadly categorized into graph-based and transition based (kubler 2009). Transition based parsers treat parsing sequence of actions that produce a parse tree and a classifier is trained to score probability actions at each stage of the process. Nivre 2008 [9] [10] provides a simple form of greedy discriminative dependency parser. Graph based parsers (McDonald, 2006) [8] treat parsing as a search-based structured prediction problem using Minimum Spanning Tree for a sentence with the goal of learning a score function over dependency trees using a Machine Learning classification.

Chenn and Manning 2014 [3] paper uses neural approach to learn dense and compact feature representations and solves the problem of sparsity of

incompleteness of the features. In this approach they represent each word as a d-dimensional dense vector (word embeddings) and part of speech tags (POS) and dependency labels are also represented as a d-dimensional vectors. They showed that neural network can accurately determine the structure of sentences, supporting meaning interpretation. Dozat and Manning 2017, Dozat, Qi and Manning 2017 [4] [5] papers propose a neural graph-based dependency parser and designed a biaffine scoring model for neural dependency parsing.

3 Implementation

Transition based Parser: For this approach we are starting with the initial conditions where stack is empty and the buffer have all the words from the sentences also, there is no relation established hence the arcs are also empty. Terminal condition is when buffer is empty and stack may or may not have words and arcs have the dependency relations.At each step we can perform any of the four transitions Left-Arc, Right-Arc, Reduce and SHIFT. The learned classifier will decide what transition to take. Following are the steps of the approach:

1. In order to generate the training data, we will parse the sentences to tuples which have word and its POS tags. The labels for the sentence are captured using the gold standard Universal dependencies.

2. For word embedding, we are going to experiment with Random Embedding, Glove embedding, Elmo embedding CSKG embedding. [12] [11] [6]

3. For the POS tags, the word2vec embeddings are taken into the model.

4. The features used for the model are top 2 words from the buffer, stack and their corresponding POS tags embedding.

5. All 4 words lemmas, distance between top words in stack and buffer in actual sentence is also taken as features.

6. We are experimenting with two different approaches for creating the features from the already established arcs:

    - average word embedding single feature from the arcs.
    - use the graph representation learning approach where each node feature are transformed based upon incoming arcs from neighboring nodes and a single feature averaged on disconnected graph from arcs.

7. These Above mentioned features are used as input for the Bi-LSTM model where at each step a single prediction from the set [SHIFT,REDUCE,LEFT-ARC,RIGHT-ARC] is selected.

8. During the training we will be applying the gold standard transition on the configuration at each step instead of prediction to avoid the initial wrong prediction on the later stages.

Graph based Parser: Following the similar approach to Dozat and Manning 2017, Dozat, Qi and Manning 2017 [4] [5] papers. For an input given sentence we represent the set of words, punctuation are represented as vertices and the head-dependent and grammatical functions relationships are represented as set of ordered set of pairs or arcs in a directed graphs G = (V, A). The parser will run the sentence through an encoder and then pass the encoded representation of two words wi and wj through a neural network that estimates a score for the edge i->j.

Data Preprocessing: First the CoNLL-U files from Universal Dependencies English Treebank are converted into json files. We add dummy 'ROOT' to the begining of each sentence in both training and test data along with a dummy POS tag (ROOT) and dummy label for arcs from ROOT to itself(-). Words that only occur once training data are replaced with <unk> and words in test data that donot occur in training data are also replaced with <unk>, since training is unlikely to accomodate such words.

Word Embedding: We then generate embedding of each word in training corpus where each word embedding vector has a dimension of 100. POS tag is also taken into consideration in determining the dependency relation, the embedding vector for POS tag is a 20-dimensional vector. The word and POS embedding are concatenated into a 120 dimensional vector. The embedding vectors are initialized using Gensim's Word2Vec method because starting with accurate embedding reduces training time. These embeddings are taken as a Parameter in the model, hence they change during training.

Model Architecture: In the model first we have a Bidirectional LSTM layer that takes input of sequence of 120 dimensional vectors similar to Kiperwasser and Goldberg(2016)[7]. The idea behind using a BiLSTM is that given a dependent's history (words that precede it) is not sufficient to determine its head, we also want to keep track of its future words. It helps in having a large context window around each dependent.

Using the approach similar to Dozat and Manning(2016) [4] we feed the output of the BiLSTM into two different feed forward networks resulting in two different output vectors. Intuitively, one of the output vector corresponds to the contextualised (word, POS tag) pair qua head while the other corresponds to a contextualized (word, POS tag) pair qua dependent.We now score the weight of an arc with j as head and i as dependent using Dozat and Manning scoring function [4] :

$score(j,i) = h_j^{(arc-head)^T} U^{(1)} h_i^{(arc-dep)} + h_j^{(arc-head)^T} U^{(2)}$ where $U^{(1)}$ is a 20x20 matrix and $U^{(2)}$ is a 20 dimensional column vector. Both are parameters of the model which are learned during training. After predicting the weights to each possible arc between two words, we then predict the label for a given arc. Following the Dozat and Manning's approach, we put each candidate head through the head MLP and each candidate dependent through the dependent MLP to obtain $h_j^{(arc-head)}$ and $h_i^{(arc-dep)}$ respectively. Then we put the 40-dimensional vector ($h_j^{(arc-head)}$ + $h_i^{(arc-dep)}$) through an MLP with single hidden layer called label classifier.

Training: for each sentence in training corpus, the model gives two outputs in each forward pass: a n x m adjacency matrix, where value at (j, i) represents the weight of an arc with head as the word indexed by j and dependent as the word indexed by i. Since each dependent has exactly on head, we can view the columns of this matrix as corresponding to a probability distribution over possible heads. Secondly it outputs a n x l matrix, where l is the number of labels in the training corpus. Each row of this matrix will correspond to a probability distribution over labels to gold arcs whose dependent is the word indexed by row.

From the gold tree for the sentence, we construct a gold n x n adjacency matrix, where non zero entries correspond to the weight of the arc in the gold tree. From gold labels for each arc, we construct a gold n x l matrix, where non zero entries correspond to the correct label for the arc. The loss is calculated using cross entropy loss (with respect to the columns) between the predicted n x n adjacency matrix and the gold n x n adjacency matrix. We also calculate the cross-entropy loss between the predicted label and gold label matrix and sum these two losses as total loss.Using this total loss we back-propagate loss through the model.

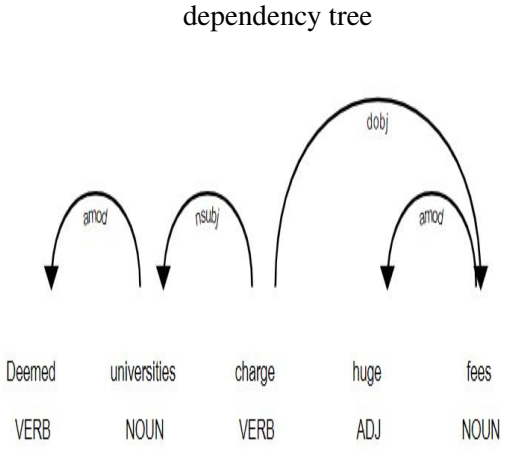

Figure 1: Resultant dependency tree with both direction and tags

4 Experimental Results

Dataset: UD Atis Treebank is a manually annotated treebank consisting of the sentences in the Atis (Airline Travel Informations) dataset which includes the human speech transcriptions of people asking for flight information on the automated inquiry systems. The data is split into 4224 training and 586 test set.[1]

Transition Based Parser: The hyper-parameters used to train the model are: word embedding size = 100, POS tag embedding size = 100, with Two Fully connected layers on top of that to Predict 3 Classes SHIFT, REDUCEL,REDUCER for 3 epochs. Due to resource constraints, we trained the network for very Few epochs. Then we further evaluated the model on test data. The Training is done using the Teacher forcing where we used the truth target labels for the transition instead of the prediction. and during inference we used both teacher forcing and using just the prediction at each step for the transitions.

Graph based Parser: The hyper-parameters

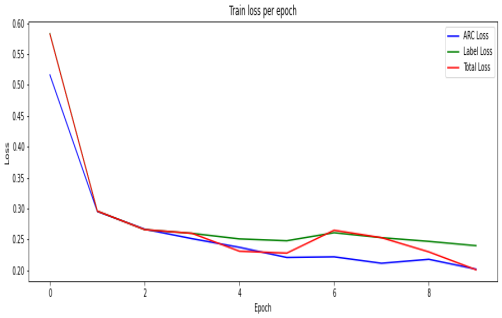

Figure 2: Train Loss of Graph based Parsing

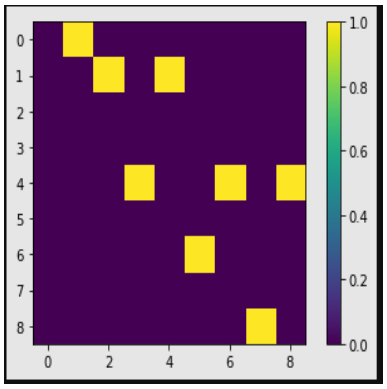

Figure 3: Heatmap for Gold Tree

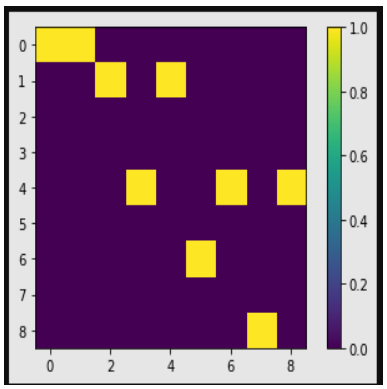

Figure 4: Heatmap of model weights after 20 epochs

used to train the model are: word embedding size = 100, POS tag embedding size = 20, BiLSTM size = 400, BiLSTM layers = 3, ARC MLP size = 500, Label MLP size = 200, weight decay = $1e^{-6}$, learning_rate = 0.001, BiLSTM dropout = 0.3, Number of epochs = 10. The heatmap of gold tree is shown in figure-3 and the heatmap of weights after 20 epochs is shown in figure-4. The train loss which comprise of arc loss , label loss and total loss is plotted in figure-2. Due to resource constraints, we trained the network for 20 epochs. Then we further evaluated the model on test data and the loss on test was: arc Loss =1.61, label loss = 1.25,total loss = 2.87. All the results and implementations for reference are shown in Ipython Notebook.

5 Conclusion

In this project we implemented transition based and neural based dependency parser. We used the Universal Dependencies tree-bank dataset to evaluate our model. For graph based parsing, first the model can be trained for more number of epochs to improve test loss without over-fitting the model.Also, further evaluations can be done in future to compare the Minimum Spanning Trees of test data and predicted labels (Gold Trees) to compute label and unlabelled arcs error scores.

References

[1] Universal dependency atis english tree-bank. https://https://github.com/howl-anderson/ATIS_dataset/blob/master/README.en-US.md.

[2] Universal dependency dataset. https://universaldependencies.org/.

[3] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[4] Timothy Dozat and Christopher D Manning. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*, 2016.

[5] Timothy Dozat, Peng Qi, and Christopher D Manning. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies*, pages 20–30, 2017.

[6] Filip Ilievski, Pedro Szekely, and Bin Zhang. Cskg: The commonsense knowledge graph. In *European Semantic Web Conference*, pages 680–696. Springer, 2021.

[7] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, 2016.

[8] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 523–530, 2005.

[9] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *LREC*, volume 6, pages 2216–2219, 2006.

[10] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

[11] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[12] Milan Straka, Jana Straková, and Jan Hajič. Evaluating contextualized embeddings on 54 languages in pos tagging, lemmatization and dependency parsing. *arXiv preprint arXiv:1908.07448*, 2019.