CENG463 - Assignment-3 Report

Ali Doğan

January 2021

1 Projective and non-Projective dependency parsing

A **Projective** dependency parsing is achieved when the arrows drawn above the words related to dependency of the words of a sentence do not contain any crossing edges. When we draw a dependency tree from parent to children edges implying the relation, then we should have a proper tree structure such that we should be able to draw the tree from root to leaves without any need of indicating the directions of the dependencies (a property of a tree structure).

In **non-Projective** dependency parsing, however, the relations do not meet the constraints above, i.e., there are at least two arrows crossing with each other when the sentence is written in ordered and relations are drawn. Examples of projective and non-projective parsings shown below.

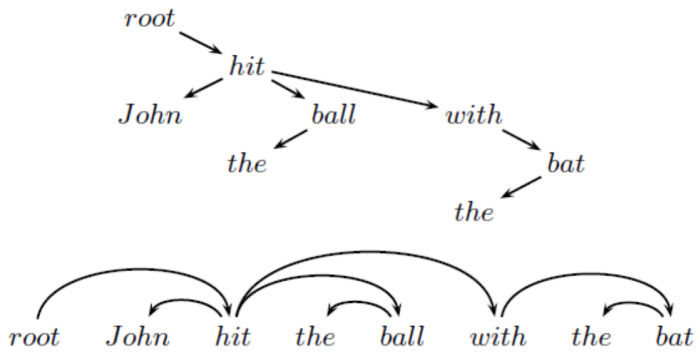

Figure 1: Projective Dependency parsing of a sentence

Figure 2: non-Projective Dependency parsing of a sentence

2 Dependency Grammars vs Context-Free Grammars

Context-free grammar, in formal language theory, is a formal grammmer with rules which implies many properties, including the independency from the context as the name suggests. In the NLP field, there are two views of linguistic structures. One is Constituency (phrase) structure and other is Dependency structure. In NLP context, Phrase structure grammars is associated with Context-free grammars.
Main differences between a dependency grammar and constituency grammars are as follow :

- Constituency grammars are syntactic grammars which is a context free grammar.
- Dependency grammars, on the other hand, are about the semantics of the sentence when we look at it from the linguistic context.

- We know that given a set of grammatical rules of a language, context free grammer can parse a sentence in polynomial time (for example via dynamic programming). For a dependency grammar, however, may not be viable due to its non-projectivity.
- A dependency grammar has a notion of a head, officially CFGs do not. However, there are parsers with the notion of a head that are used (Charniak,Collins,Stanford,..). There are formal definitions on the context of this comparison that can be looked in more detail. (Abney,1994)

3 English vs Turkish language

In the comparison between English and Turkish about the dependency parsing and projectivity, morphology of the languages plays an important role. We see a huge difference between English and Turkish in the structure. In both languages, the root of the sentence, hence the root of the dependency parsing, is usually the verb. However, the location of the root in these two languages is different. Turkish is predominantly (but not exlusively) head final(Eryiğit et al.,2008). In English, however, the head of the sentence is at the middle.

Turkish language is a very rich agglutinative language with a free constituent order and showing head final structure as mentioned above. Two properties are important in the projectivity context.

- Constituents may freely change their position, this may cause non-projectivity.
- Turkish words can be formed through derivations. Nouns can give rise to many inflected forms and verbs.

Due to the latter property, having a dependency relation only among the words is not preferred. This might be the biggest difference between English and Turkish dependency parsing. There are many works showed such morphological structure of Turkish by splitting the words into inflectional groups(Eryiğit et al.,2008). A word is divided into Inflectional groups (IGs), which are root and derivational elements of the word. A dependency relation structure of a sentence with splitting the words into IGs is shown with an example below:

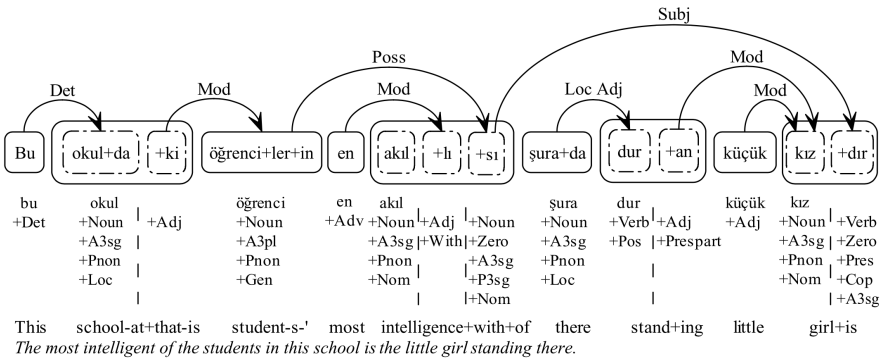

Figure 3: Dependency relations in an example from (Eryiğit et al.,2008)

Since the head of the sentence is usually at the end, Turkish language may seem to have less non-projective instances compared to English at first. However, due to the properties explained above, we may conclude it is the opposite.
We may be also get some insight by looking at the question sentences. In Turkish, the question sentences are constructed by mostly substituting the answer by the question word. (e.g., "Ali ata baktı" turning into "Ali neye baktı?" do not cause any problem for projectivity). However, if we look at the English, it is not always like this. (e.g., "Bill buy the coffe from Ali yesterday" turning into "Who did Bill buy the coffee from yesterday" causes the latter sentence being non-projective)

3.1 UD English-ParTUT vs UD Turkish-IMST

SpaCY provides very powerful commands to investigate the conll documents thorough terminal.

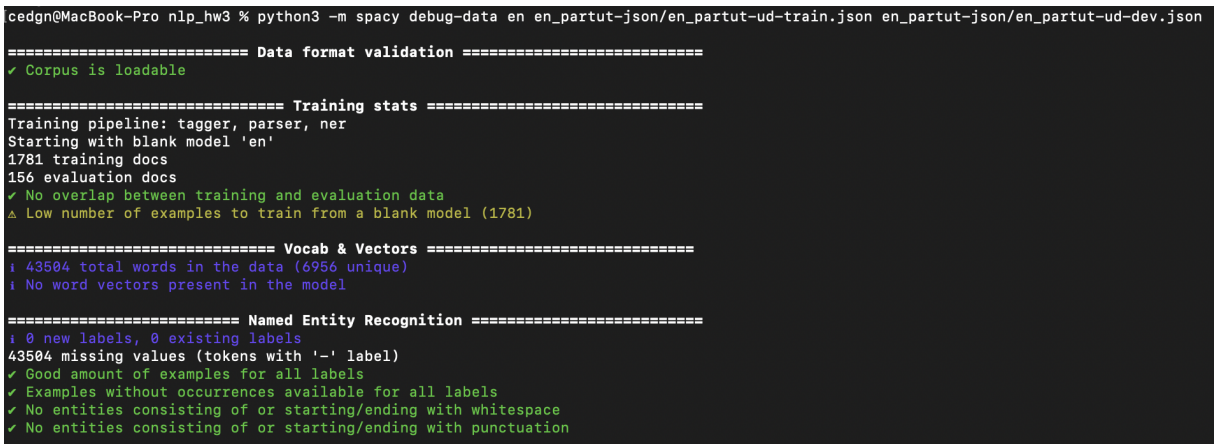

Figure 4: Data statistics of UD-English parTUT

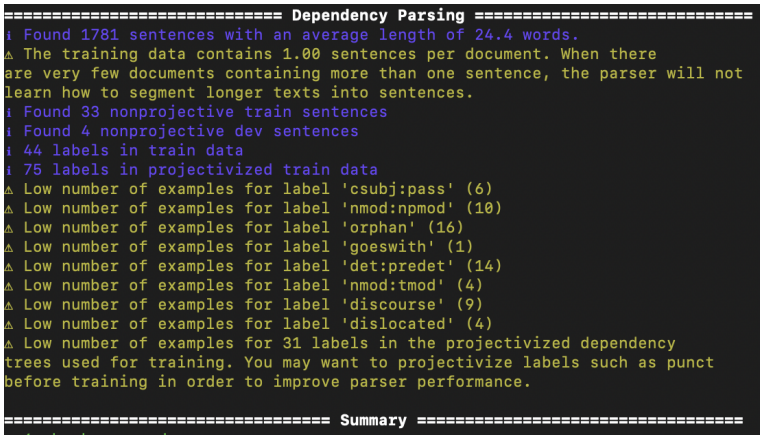

Figure 5: Dependency Parsing statistics of UD-English parTUT

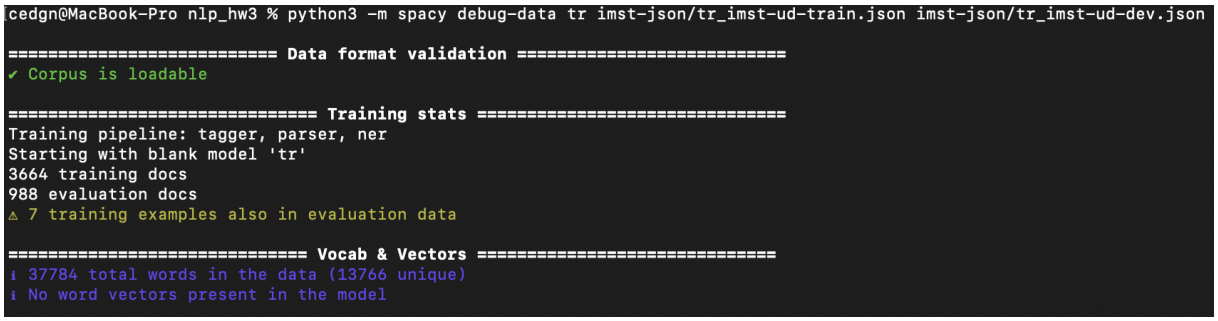

Figure 6: Data statistics of UD-Turkish IMST

```
========================== Dependency Parsing ==========================
ℹ Found 3664 sentences with an average length of 10.3 words.
⚠ The training data contains 1.01 sentences per document. When there
are very few documents containing more than one sentence, the parser will not
learn how to segment longer texts into sentences.
ℹ Found 405 nonprojective train sentences
ℹ Found 119 nonprojective dev sentences
ℹ 32 labels in train data
ℹ 209 labels in projectivized train data
⚠ Low number of examples for label 'vocative' (2)
⚠ Low number of examples for label 'parataxis' (10)
⚠ Low number of examples for label 'advcl' (4)
⚠ Low number of examples for label 'csubj' (8)
⚠ Low number of examples for label 'dep' (1)
⚠ Low number of examples for 174 labels in the projectivized dependency
trees used for training. You may want to projectivize labels such as punct
before training in order to improve parser performance.
```

Figure 7: Dependency Parsing statistics of UD-Turkish IMST

We see that the English treebank has 1781 docs in the training file, 156 in the dev file. On the other hand, Turkish IMST treebank has 2664 docs in the training file, 988 in the dev file. We also see that the length of the English sentences in the ParTUT is larger than in the Turkish treebank (24.4 vs 10.3). This could be due to different sources of data or could be due to the fact that Turkish is agglunitive language which makes it more compact in lengthwise.
In the dependency parsing statistics part, we see 33 non-projective sentences in English, 405 in Turkish. If we divide them into the total number of sentences we get the followings:

$$\textbf{non-projectivity ratio in Turkish-IMST} = \frac{405+119}{3664+988} = 0.112$$

$$\textbf{non-projectivity ratio in English ParTUT} = \frac{33+4}{1781+156} = 0.019$$

From the results, we see that roughly 2 percentage of the documents is non-projective in English ParTUT treebank whereas in Turkish-IMST, we have 11 percentage of the documents displaying non-projectivity.

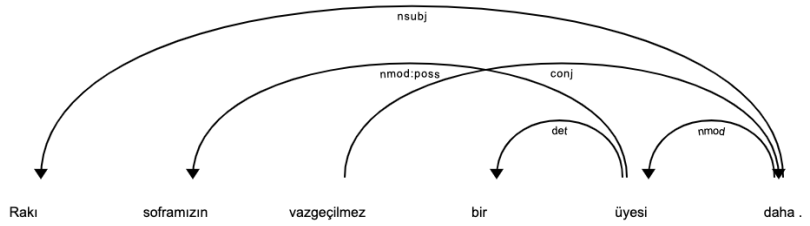

Figure 8: non-Projective parsing example from Turkish-IMST Treebank

4 Transition-based and Graph-based Dependency parsing

Transition-based parsers construct the dependency trees by going left-to-right or right-to-left thorough the words of a sentence. And, each time, the parser selects an action decided by the rules of the system. One such example is MaltParser(Nivre et al., 2006).

Graph-based parsers are a top-down approach to find the optimal solution. They parametreize a model over smaller substructures in order to search for a valid dependency graph and produce the most likely one. (McDonald and Nivre, 2007)

5 LAS and UAS attachment scores

In the dependency parsing task, two features are predicted: the syntactic head, and the dependency label.

Labeled Attachment Score (LAS) is a standard evaluation metric in dependency parsing. It is calculated as the percentage of words that are assigned **both** the correct syntactic head and the correct dependency label.

Unlabeled Atachment Score (UAS) , on the other hand, is calculated as the percentage of words that are assigned the correct syntactic head without taking into account the dependency label. Which implies UAS is always greater than or equal to LAS score.

6 Dependency Parser İmplementation

I have implemented a Dependency Parser using SpaCy's command line for training followed by the model usage in python3. SpaCy provides a training of a specified model thorugh command line after converting the conllu format data into a json format.

```
git clone https://github.com/UniversalDependencies/UD_Spanish-AnCora
mkdir ancora-json
python -m spacy convert UD_Spanish-AnCora/es_ancora-ud-train.conllu ancora-json
python -m spacy convert UD_Spanish-AnCora/es_ancora-ud-dev.conllu ancora-json
mkdir models
python -m spacy train es models ancora-json/es_ancora-ud-train.json ancora-json/es_ancora-ud-dev.json
```

Figure 9: Training commands from SpaCy

Trained models on each epoch is saved under the given directory. The saved models can be easily loaded by SpaCy.load("model_path function provided in SpaCy module in python. It is also possible to adjust hyper parameters to optimize the model. The parameters with their default values are providedin SpaCy documentation.

I have tried couple of different dropout rates, learning rate and hidden layer size. I have not achieved a significant difference in the UAS/LAS accuracy of dev set of different training models.

After hyper parameter optimization part, I have pursued on the best model I achieved by testing it on the test set. Best model was from the following configuration of the training :

```
cedgn@MacBook-Pro nlp_hw3 % alias train-parser="python3 -m spacy train tr models_pt_lr003  imst-json/tr_imst-ud-train.json imst-json/tr
_imst-ud-dev.json -p 'parser' -n 30 -pt 'dep' "
cedgn@MacBook-Pro nlp_hw3 % dropout_from=0.4 dropout_to=0.4 hidden_width=256 learn_rate=0.003 train-parser
```

Figure 10: Training configuration of the best model achieved

Adding a side objective to the CNN model of the training improved the model to learn faster. Moreover, increasing the learning rate along with adding more neurons in the hidden layer and increasing dropout rate contributed to the accuracy with at least +1 percent.

The UAS/LAS versus iteration plot of the best model is as follow:

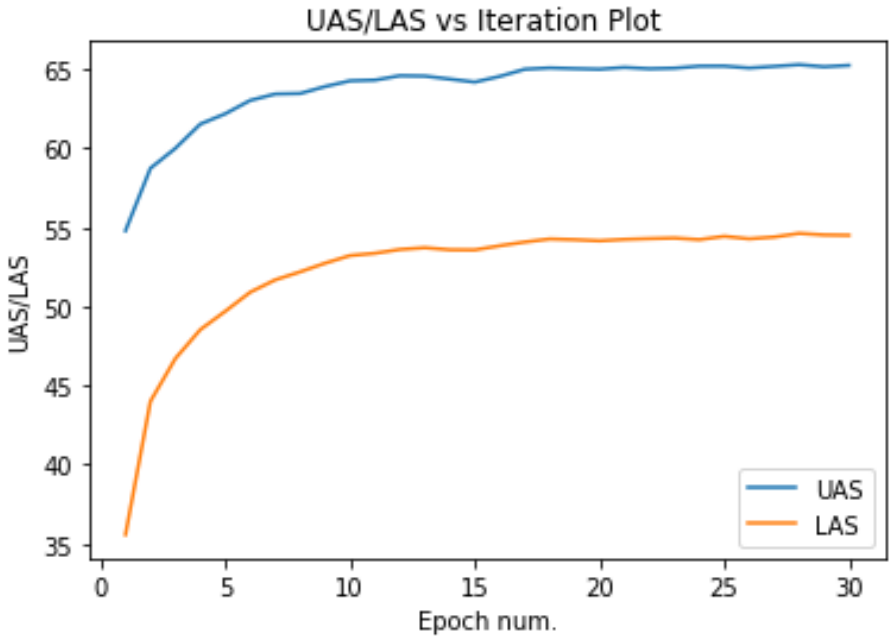

Figure 11: UAS-LAS vs iteration plot of the best model achieved

The attachment scores above are of the development dataset. The test set evaluation of the model is checked via SpaCy command line as below :

```
cedgn@MacBook-Pro nlp_hw3 % python3 -m spacy evaluate models_pt_lr003/model-best imst-json/tr_imst-ud-test.json  -dp eval_result_dep1 -
R

================================== Results ==================================

Time      0.84 s
Words     10029
Words/s   11881
TOK       100.00
POS       0.00
UAS       65.54
LAS       54.75
NER P     0.00
NER R     0.00
NER F     0.00
Textcat   0.00

✔ Generated 25 parses as HTML
```

Figure 12: Evaluation of the model on test set.

As seen, I have achieved a 54.75 LAS accuracy on the test set. It is very close to the development set evaluation above, implying that the model is generalized well. In LAS Ranking of CoNLL 2018 Shared Task, the accuracy I achieved is in the 24th placement, just above the HUJI (Yerushalayim) model.

The evaluate command also generates a number of random parsed sentences by the model. One example is below:

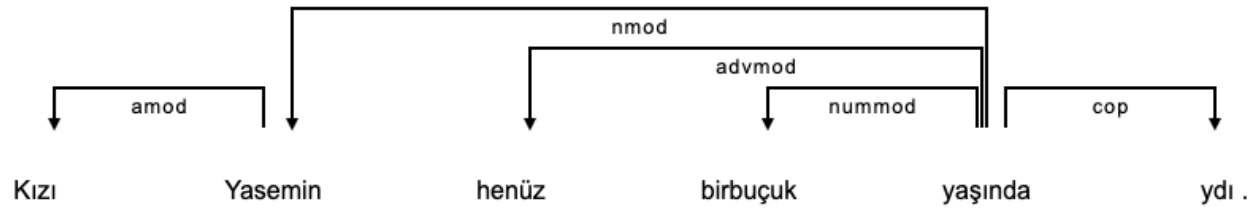

Figure 13: A random example of dependency parsing from test set.

A script that loads the trained model and shows how to use the model by parsing the entire conll document and printing the resulted parses into conll is provided. Also one can use the model to parse arbitrary generated sentences by loading the model which provided along with the report.

7 Conclusion

In this assignment, dependency parsing properties are investigated and dependency parser on Turkish language is conducted by training it on Turkish-IMST Treebank. SpaCy software package is used for the tasks above. The resulted parser model has achieved a 24th place in the CoNll Shared task ranking with a LAS score of 54.75 on the test set.

We see in an example from test set, the splitting of the words into its inflectional forms("yaşında" - "ydı"). However, one must note that it is not a complete split. I have realized this issue while looking at the input and output conllu formats. The input file does not contain detailed inflectional group parsing. Also, the base Turkish language properties provided by SpaCy may not be sufficient to parse such groups. I believe such refinements will improve the accuracy of the model on attachments scores significantly.

8 References

- Alicia Ageno, "Statistical Processing of Natural Langeage: Dependency Parsing", `https://www.cs.upc.edu/~horacio/snlp/depparsing-snlp.pdf`
- Steven Abney, 1994, "Dependency Grammars and Context-Free Grammars", `http://www.vinartus.net/spa/94g.pdf`
- Eryiğit, Gülşen & Nivre, Joakim & Oflazer, Kemal. (2008). Dependency Parsing of Turkish. Computational Linguistics. 34. 627. 10.1162/coli.2008.34.4.627.

- Nivre, Joakim & McDonald, Ryan. (2008). Integrating Graph-Based and Transition-Based Dependency Parsers.. 950-958.
- Nivre, Joakim & Hall, Johan & Nilsson, Jens. (2006). MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. Proceedings of LREC.