Assignment - 3

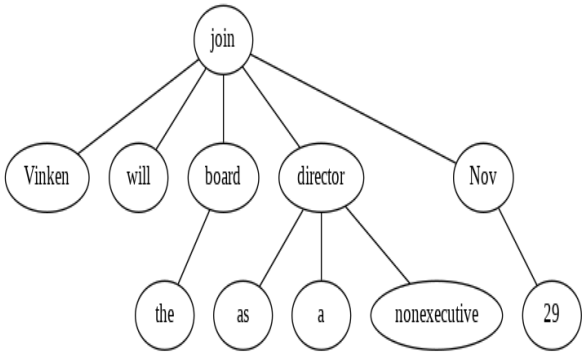
160050012-Piyush Onkar
203050083- Sandeep Parihar
203050104-Vishal Pramanik

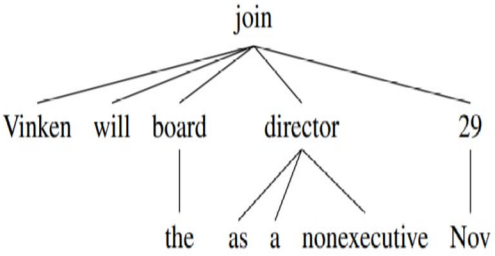
1.CP to DP Conversion

A common way to find the head in a phrase structure is to use a head percolation table.
Once the heads in phrase structures are found, the conversion from phrase structures to dependency structures is straightforward, as shown below:
(a) Mark the head child of each node in a phrase structure, using the head percolation table.
(b) In the dependency structure, make the head of each nonhead-child depend on the head of the head-child.

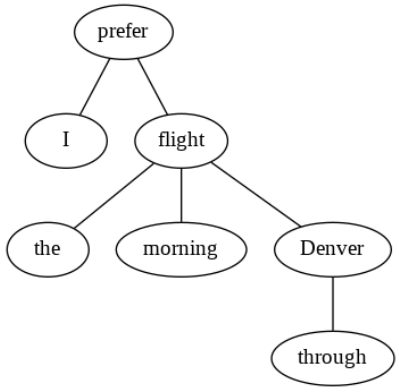
Inputs to our parser are obtained from the Allennlp.org (Allennlp parser).

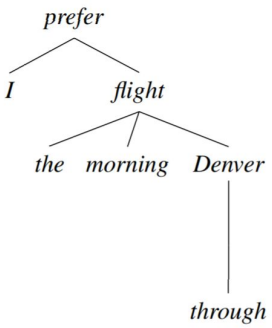
Input sentence : Vinken will join the board as a nonexecutive director Nov 29





Input sentence : I prefer the morning flight through Denver





Converted Graph Original Graph

We are able to generate the correct Dependency Tree.

Error Analysis

Unlabelled Dependency Tree

Our approach works fine for getting unlabelled dependency tree.

We are getting almost similar tree when compared with the output of Spacy,Stanford and Allennlp parser.

We have tested on many examples and very few of them vary from the actual output.

Since we have used simple rules our parser may not gives most accurate answer for the long dependency structures.

Labelled Dependency Tree

We have used the labelling algorithm based on the research paper {Guidelines for the Clear Style Constituent to Dependency Conversion} by Jinho D. Choi and Martha Palmar.

Since parsers like Spacy ,Allennlp does not provide function tags and this algorithm requires function tags.

We are able to assign the basic labels.

Below is the Dependency Labelled Tree which we are getting.

```
***Dependency Tree***

playing => Students     >> playing {VMOD} Students
playing => are  >> playing {AUX} are
playing => well >> playing {VMOD} well
well => very    >> well {ADVMOD} very
```

Universal dependencies

```
nsubj(playing-3, Students-1)
aux(playing-3, are-2)
root(ROOT-0, playing-3)
advmod(well-5, very-4)
advmod(playing-3, well-5)
```

References :

Converting Dependency Structures to Phrase Structures:
Fei Xia and Martha Palmer

Guidelines for the Clear Style Constituent to Dependency Conversion
Jinho D. Choi & Martha Palmer

2. DP to CP Conversion

We are using spacy library for converting sentence to dependency parse Tree.

e.g) The boy is eating with a hand

The dependency parse Tree using using spacy :

The => det => boy,
boy => nsubj => eating,
is => aux => eating,
eating => ROOT => eating,
with => prep => eating,
a => det => hand,
hand => pobj => with .

We developed some rules for converting from dependency parse tree to constituency parse tree but there is problem in conversion. We are currently working on the algorithm and adding few more rules.

Error Analysis

Dependency Parse Tree :

The => det => boy,
boy => nsubj => eating,
is => aux => eating,
eating => ROOT => eating,
with => prep => eating,
a => det => hand,
hand => pobj => with

The correct constituency parse tree should be :
(S (NP (DT The) (NN boy)) (VP (VBZ is) (VP (VBG eating) (PP (IN with) (NP (DT a) (NN hand))))))

but the algorithm is returning parse tree as
(S (NP(The DT)(boy NN))(VP(VP(is AUX)(eating VPStarting))(PP(with IN))(NP(a DT)(hand NN)))

Code Repository Explanation :

We have written all code in Jupyter Notebook. The code has to be run on Google Collab so that all libraries and dependency files are automatically downloaded.

To get the inputs for the parse Use the following code:

```
pip install allennlp==1.0.0 allennlp-models==1.0.0
from allennlp.predictors.predictor import Predictor
import allennlp_models.structured_prediction
predictor =
Predictor.from_path("https://storage.googleapis.com/allennlp-public-models/elmo-constituency-parser-2020.02.10.tar.gz")
predictor.predict(
  sentence="If I bring 10 dollars tomorrow, can you buy me lunch?"
)
```

Or simply get the inputs from the Stanford parser :

http://nlp.stanford.edu:8080/parser/