CS626 A3 - Transforming Parse Trees

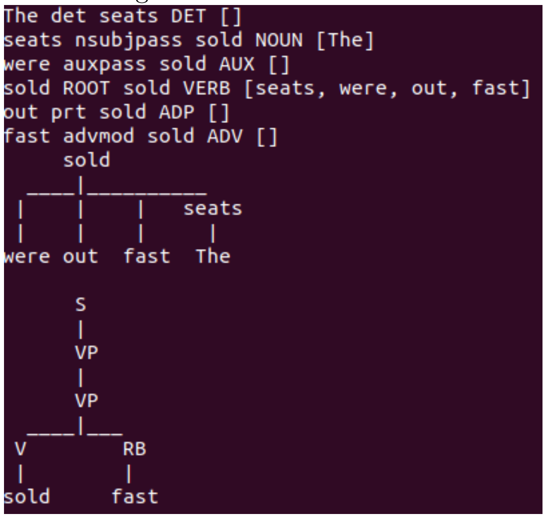
Shreya Pathak, Neel Aryan Gupta, Mohammad Ali Rehan

180050100, 180050067, 180050061

1 Dependency parse to Constituency parse transformer

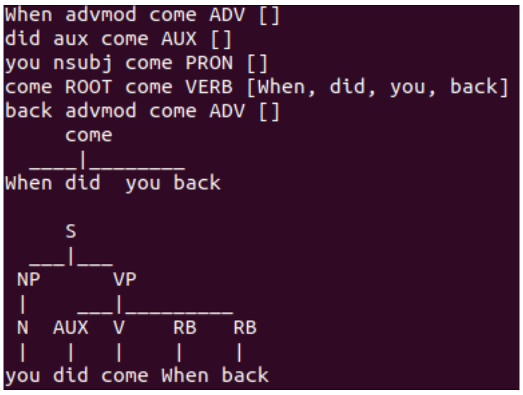
The transformer is a complete functional tool that takes as input a sentence, generates a dependency parse and applies our top down algorithm to get the constituency parse. The code base is complete for both the tools.
We use the generic grammer given below to represent the CP tree and find these corresponding phrases from the dependency tree.
S⟶ NP VP SBAR
NPS⟶DET JJP NNS/NNP SBAR | (NP CONJ NP)+
VPS⟶AUX* NEG* VBD/VBG ATTR* NP1* NP2* ACOMP* RB* PP* SBAR* | (VP CONJ VP)+
JJPS⟶JJ+ | (JJP CONJ JJP)+
PPS⟶ IN NP
SBARS⟶WHWORD S
* means that the particular phrase might not be needed or may be more than one in number, + means that the phrase will be at least one in number

Drawbacks:

1. Passive voice sentences are not handled properly as they involve tags like nsubjpass etc which we have not encoded. Eg: The seats were sold out fast


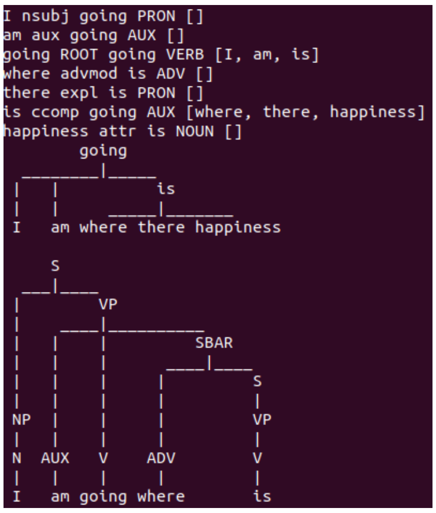
```
The det seats DET []
seats nsubjpass sold NOUN [The]
were auxpass sold AUX []
sold ROOT sold VERB [seats, were, out, fast]
out prt sold ADP []
fast advmod sold ADV []
     sold
  ____|__________
 |    |    |   seats
 |    |    |     |
were out  fast  The

      S
      |
      VP
      |
      VP
  ____|___
 V       RB
 |        |
sold     fast
```

2. Questions are not handled properly. Eg: When did you come back?



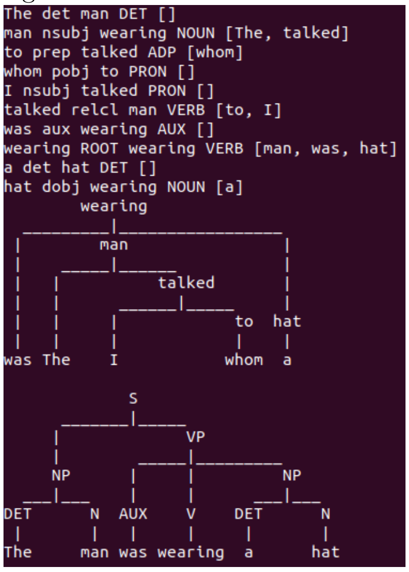
```
When advmod come ADV []
did aux come AUX []
you nsubj come PRON []
come ROOT come VERB [When, did, you, back]
back advmod come ADV []
     come
  ____|________
When did  you back

     S
  ___|___
 NP      VP
 |    ___|_________
 N  AUX  V    RB   RB
 |   |   |     |    |
you did come When back
```

3. The identification of label of whword and labels in its subtree is not always correct because of the various roles it can take in the clause like object, subject etc.
   Eg:I am going where there is happiness



```
I nsubj going PRON []
am aux going AUX []
going ROOT going VERB [I, am, is]
where advmod is ADV []
there expl is PRON []
is ccomp going AUX [where, there, happiness]
happiness attr is NOUN []
        going
  ________|_____
 |   |          is
 |   |     _____|_______
 I   am where there happiness

     S
  ___|____
 |        VP
 |    ____|__________
 |   |    |          SBAR
 |   |    |       ____|____
 |   |    |      |         S
 |   |    |      |         |
 NP  |    |      |         VP
 |   |    |      |         |
 N  AUX   V     ADV        V
 |   |    |      |         |
 I   am going where        is
```

Eg: The man to whom I talked was wearing a hat



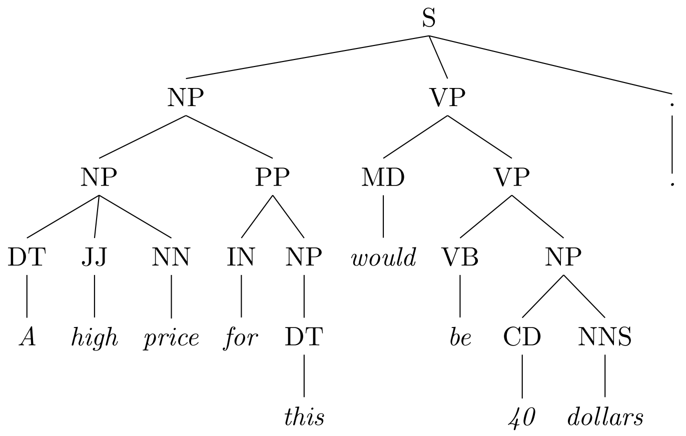
```
The det man DET []
man nsubj wearing NOUN [The, talked]
to prep talked ADP [whom]
whom pobj to PRON []
I nsubj talked PRON []
talked relcl man VERB [to, I]
was aux wearing AUX []
wearing ROOT wearing VERB [man, was, hat]
a det hat DET []
hat dobj wearing NOUN [a]
        wearing
  _________|_________________
 |         man               |
 |    _____|______           |
 |   |           talked      |
 |   |     ______|_____      |
 |   |    |            to  hat
 |   |    |            |    |
was The   I           whom  a

             S
      _______|_____
     |             VP
     |        _____|_________
     NP      |     |         NP
  ___|___    |     |      ___|___
DET      N  AUX    V    DET      N
 |       |   |     |     |       |
The     man was wearing  a      hat
```

2 Constituency parse to Dependency parse transformer

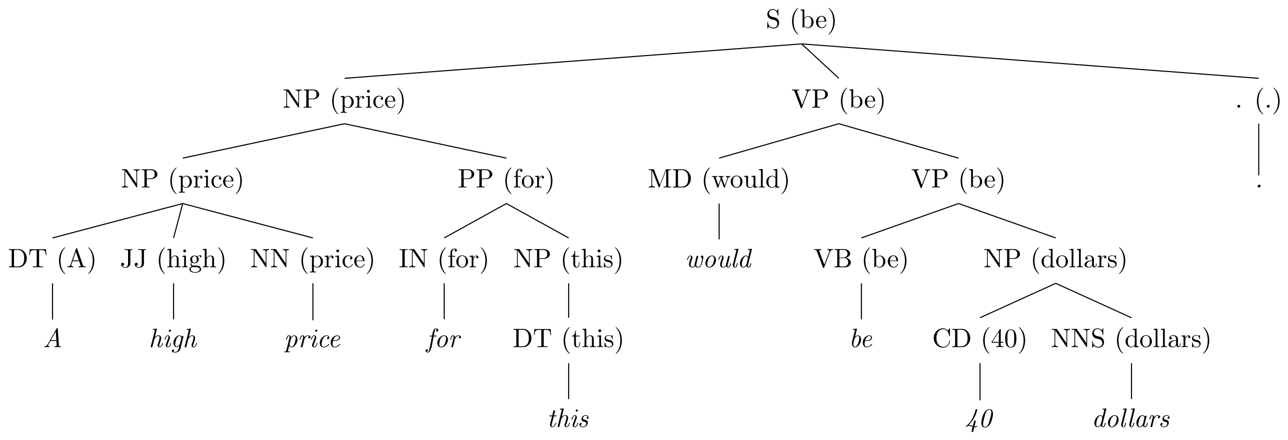
This model uses the 'benepar' library as a pipeline to SpaCy to get the constituency parse of the sentences on-the-go, and converts those parses to typed dependencies along with the dependency tree and uses NLTK Tree for proper visualisation.

The algorithm used for the transformation is based on Xia and Palmer (2001) [1].

Here is an example of a constituency parse tree obtained by the parser.



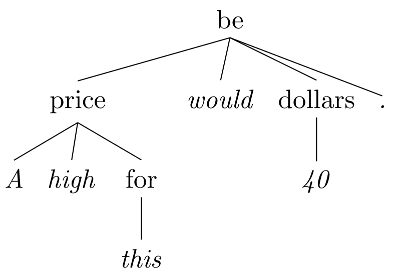
Lexical heads are a concept/idea in linguistics wherein each context-free rule has associated with it a "special" child (the lexical head) which is a central sub-constituent of the rule. It is basically one word which represents the crux of the sentence. Firstly, we assign lexical heads at each level of the tree using the head rules suggested by Collins (1999) [2]. The rules are a modified version of those used in the SPATTER parser by Magerman 1995 [3]. Here is how the tree with assigned lexical heads (referred to as 'headified' tree) looks like:

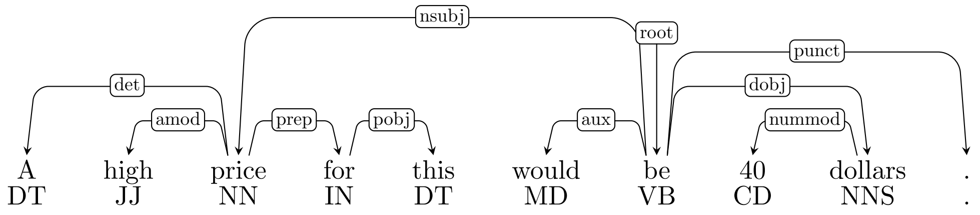


After the heads are assigned, the head of each non-head child is made to depend on the head of the head-child in the dependency structure. As an example, for the particular aforementioned tree, *price* (non-head child) depends on *be* (head child), *this* depends on *for* etc.

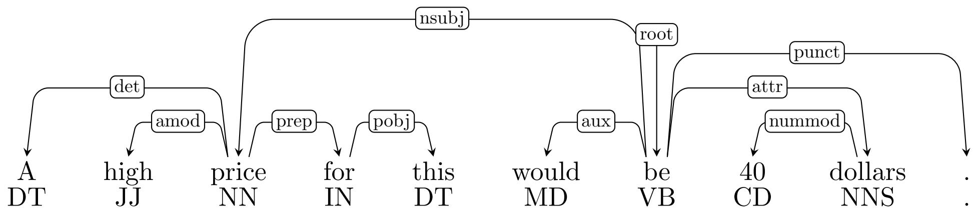
The model uses local information to determine the edge labels in the resulting dependency tree. The local information comprises of the phrase chunk labels of the lexical heads at the ends of the given edge, along with their POS tags. In some complex cases, this information is not enough to determine the edge label, for example, edge labels for passive sentences, coordinating conjunctions, but for many sentences, most of the basic modifiers are correct.

Here is the dependency parse given by the above algorithm.

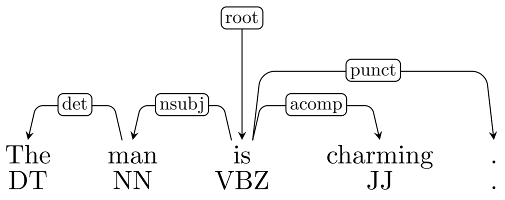


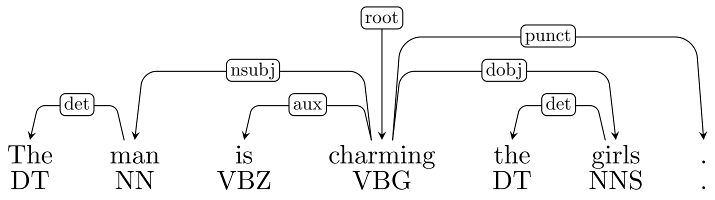


Here is the correct dependency parse :



Overall, the model performs fairly well. An example of the discrimination capability of the model with two similar examples :

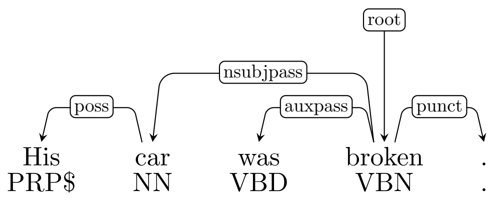


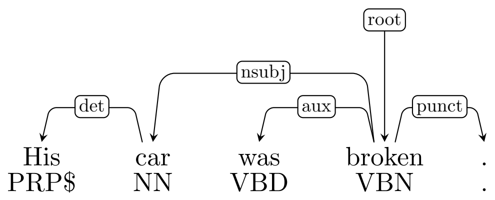


Note that the roots are correctly recognised in both the cases.

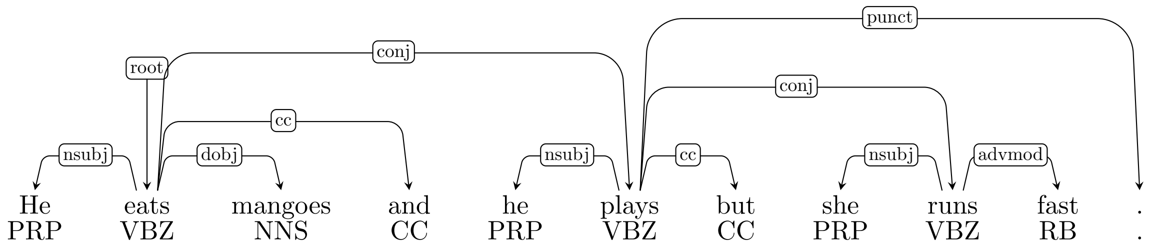
Some drawbacks of the model are as follows:

- Passive voice sentences for which the modifiers should ideally be marked as 'nsubjpass', 'auxpass' etc. are currently not handled by the model. For Example, the correct and the produced parses are respectively :

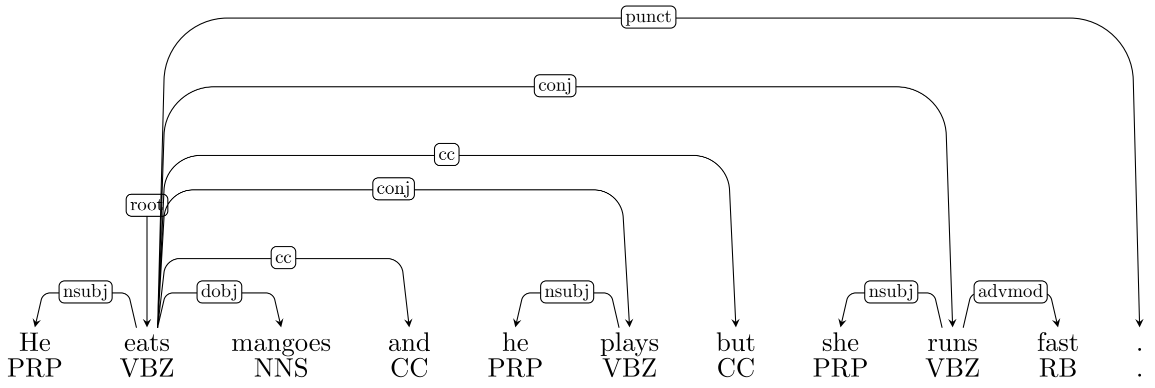




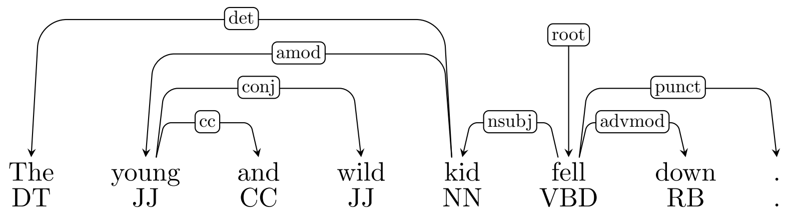
- Sentences involving multiple conjunctions are not handled properly, depending on the type of the phrase-chunks obtained from the parser. For example, here are the correct and the produced parses respectively.



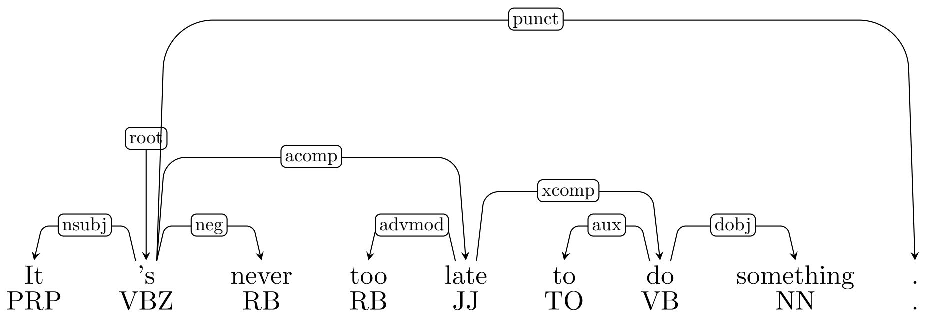
The model failed to associate the verb *runs* with the conjunction *but* resulting in the following parse tree.

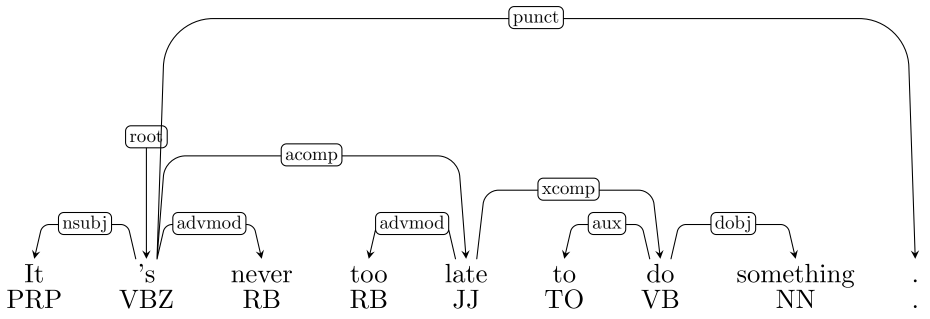


- Some complex tags such as 'dative', 'attr' (see the given example) are mispredicted because prediction of these relations requires much more information than just the local tags.
- 'prt' modifiers are mistaken as 'advmod' because this relation excludes literal/directional uses of prepositions/particles, in which case 'advmod' is used as the relation. Example :

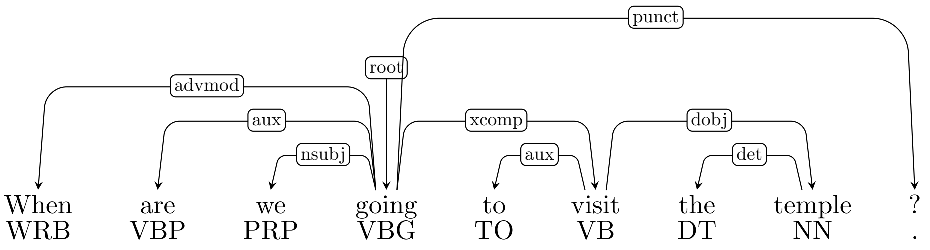


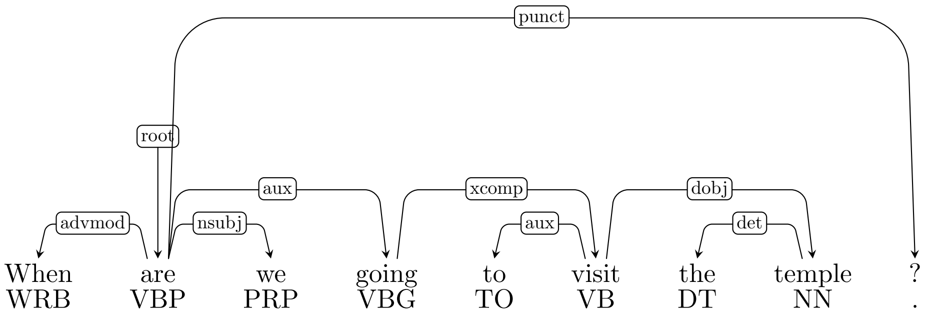
  Here 'prt' should be the relation instead of 'advmod'.
- Model fails to identify the 'neg' modifier, because the negative words like *not*, *never* are usually tagged as 'RB', and it's hard to deduce such words from the tags themselves, but can be done with hardcoding all the negative words from a large enough corpus.





- The model sometimes fails to identify the verbs properly because the fixed set of head-rules used in the model are not enough to capture all the ambiguities of the language. For example, here are the correct and the produced parses respectively.





References

[1] Xia, F. and Palmer, M. (2001). Converting dependency structures to phrase structures. In HLT-01, 1–5.

[2] Collins, M. (1999). Head-Driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia.

[3] Magerman, D. M. (1995). Statistical decision-tree models for parsing. In ACL-95, 276–283.