L95 Practical Report

Comparison of the Stanford Parsers

Zébulon Goriely, Queens', zg258

Word Count: 4973[1]

1 Introduction

In the last two decades, statistical parsers have reported increasingly competitive scores and a number of different algorithms have been shown to be effective for the task. In this report, I examine three of the Stanford parsers and evaluate them on a set of 25 challenging sentences, for which I produced gold standards. Comparing parsers to the gold standards, I find that the Neural parser outperforms both the Unlexicalised parser and the Shift Reduce parser. I also find that there are certain constructions that all parsers consistently fail to parse. I evaluate these findings quantitatively by running MaltEval, confirming my findings by comparing the precision, recall and F_1 scores for each dependency label. Finally, I discuss my evaluation methods and state what further work must be done.

2 Background

2.1 Universal Dependencies

One possible output of a parser is a **dependency tree** or a **dependency graph**. These are structures that capture the grammatical relations in a sentence, such as the object and subject of a predicate. They can be more informative than phrase structure trees for the task of automatic text understanding, particularly for providing information about predicate-argument structure, which may not be not directly available from the phrase structure. For example, the constituency parse of "I gave Kim a cat" may label "Kim" and "a cat" as two noun phrases (NPs) contained within the main verb phrase (VP). This representation does not allow us to distinguish the direct object of "give" from the indirect object. A dependency tree represents these relationships directly as seen in **??**.

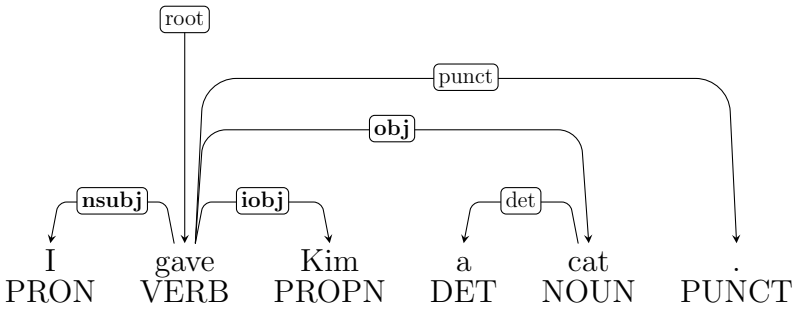


Figure 1: Example of a Dependency Parse using Universal Dependencies and Universal POS tags with predicate-argument structure relations in **bold**.

The Universal Dependencies (UD) project (**??**) was developed to provide a universal set of dependencies that can be used to capture grammatical relations across many languages. In 2015 the Stanford

[1] *texcount report.tex*

parsers switched to the UD scheme for their outputs, having previously used the Stanford scheme (**?**). Here, I consider the UD scheme.

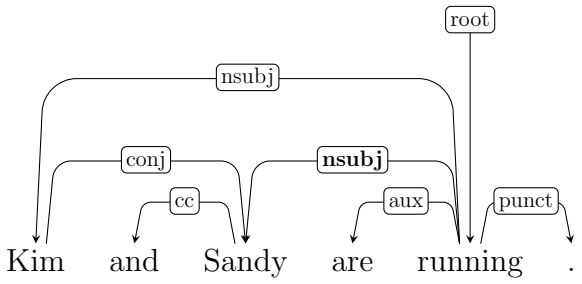


Figure 2: Enhanced Universal Dependencies parse with enhanced dependency in **bold**.

The UD scheme also provides an *enhanced* representation. The default representation produces UD trees with directed edges. Despite the computational advantages of using a tree structure, there are situations where we may want to detect relations that are not encoded by the default representation, possibly requiring the traversal of long dependency paths to find. The *enhanced* representation produces a UD *graph* that renders many of these relations explicit. A simple example of its usefulness can be seen in **??**, where we would otherwise need to traverse the **conj** dependency to find the second subject of "running". The Stanford parsers support the enhanced scheme, but here I only consider the basic representation as it is more interesting to see how the parsers fail on these basic relations rather than whether they successfully add additional information as a post-processing step.

2.2 Stanford Parsers

The Stanford NLP Tools software provides six different parsers:

1. Unlexicalised PCFG parser
2. Lexicalised dependency parser
3. Factored model (combines estimates of unlexicalised and lexicalised)
4. TreeRNN parser
5. Shift-reduce constituency parser
6. Transition-based neural dependency parser

Some of these parsers produce dependencies directly (lexicalised, neural) and the others produce constituency trees, which can be later converted to dependency trees using a conversion script. In this report, I compare the Unlexicalised parser, the Shift Reduce parser and the Neural parser, examining the UD trees that they produce. These three parsers represent a variety of approaches to statistical parsing so it is interesting to compare their performance on challenging inputs.

The Unlexicalised parser was produced by **?** as a response to the conviction at the time that lexicalised probabilistic context-free grammars (PCFGs) were an important tool for increasing the performance of probabilistic parsers, in particular for resolving syntactic ambiguities such as PP attachment (**?**). The authors discuss the advantages of unlexicalised grammars for computational efficiency and ease-of-use, and show that by using a variety of methods orthogonal to or simpler than lexicalisation (such as *parent annotation*), their parser could achieve better performance than some lexicalised models

(though not better than state-of-the-art). The aim of the paper was not to disregard the need for lexicalised models, but to show that their importance may have been overstated due to a lack of a strong unlexicalised PCFG parser baseline. The parser uses a basic CKY algorithm, operating in $O(n^3)$, using maximum-likelihood estimation for rule-probabilities. The increase in performance was achieved by adjusting the features (tag splitting, parent annotation, markovisation, head annotation), rather than the algorithm.

The Shift-reduce parser (**?**) was added to the Stanford parsers package in 2014, based on a collection of papers (**?????**). Following the success of transition-based algorithms for building dependency trees, it produces phrase structure trees by applying a set of transitions (including shift and reduce) to a queue of words, storing a partially completed tree on a stack until the parse is complete. Transitions are selected by a perceptron that uses features of the current state of the system as input, trained by iterating over trees and adjusting weights when mistakes are made. This is a very different approach to the unlexicalised parser, which uses dynamic programming with weights corresponding to grammar rules, rather than transitions, to find the best parse. Two models for the parser are provided for English, one that uses greedy transitions and another that uses a beam search of size 4. The site claims that both variants are faster than the Unlexicalised parser and achieve higher F1 scores on section 23 of the Wall Street Journal section of the Penn treebank. In this report, I investigate the performance of the greedy model.

The Dependency parser (**?**) also uses a transition-based/shift-reduce algorithm but rather than producing phrase structure trees, it produces dependency trees. A neural network is trained that selects one of three transitions (left-arc, right-arc and shift) given a dense vector representation of words, part-of-speech (POS) tags and dependency labels. This vector is computed from the current configuration of the system, which consists of a stack for the heads of subtrees, a buffer for unvisited words and a set of recorded dependencies. This parser also uses greedy decoding (beam search is left to future work). Given that this system is purely discriminative and does not generate phrase structure trees or make use of any grammar, it will be interesting to compare how it performs for complex linguistic constructions compared to the other two parsers.

3 Method

To evaluate the performance of the parsers, I was provided with a small set of 25 sentences of increasing length and complexity, containing a variety of constructions that parsers may struggle with. These sentences were formatted in a PDF file, making use of italics and other symbolic features that are not supported by the parsers. In order to convert these sentences into a plain-text format for the parsers, I first copied-and-pasted the text from the PDF into a plain-text file. I then wrote a pre-processing script `pre-processor.py` that removed page numbers and end-of-line hyphenations.

The slightly simplified version of the Viterbi algorithm that we present takes as input a single HMM and a sequence of observed words $O = (o_1, o_2, \ldots o_T)$ and returns the most probable state/tag sequence $Q = (q_1, q_2, q_T)$ together with its probability.

(a) Original Formatting.

```
The slightly simplified version of the Viterbi algorithm
that we present takes as input a single HMM and a sequence
of observed words O = (o1, o2, ...oT ) and returns the most
probable state/tag sequence Q = (q1, q2, qT ) together with
its probability.
```

(b) After Pre-Processing.

Figure 3: Pre-Processing Sentence 18.

This process did remove information that could be useful for parsing, such as formatting and mathematical symbols. One example of this can be seen in **??**. Here, the subscripted numbers have been converted to full-size numbers and the italics has been lost. The loss of this information has a number of consequences, in particular for the *use/mention distinction* problem discussed in **??**.

To run the parsers, I used the The Stanford CoreNLP package[2]. Given raw input text, a pipeline performs a series of configurable operations including tokenisation, sentence splitting, part-of-speech tagging and parsing. By using this pipeline and only adjusting the parsing step by selecting the relevant parser, I was able to ensure that all three parsers operated on the same sentence representations.

I wrote a script `parse_all.sh` to parse the sentences using the three parsers with default parameters. I outputted the parses in both plain-text and Conll-U formats. The former allowed me to quickly read the output of the parsers and the latter allowed me to transfer the parses to visualisation software for analysis and correction.

Then, I performed qualitative analysis of the parsers by comparing the output of the three parsers for each sentence. This step was performed by copying the parses to UD Annotatrix[3], a basic editor for Universal Dependency trees. For each sentence, I corrected the parses to create gold standards, referencing the Universal Dependencies manual (**?**). To improve my gold standards, I collaborated with another student, Alodie Boissonneat, comparing my set of gold standards with hers and sharing linguistic interpretations to find mistakes and discuss ambiguities. I then compared the revised gold standards to the parses, noting where the three parsers made mistakes. The results of this process are discussed in **??**.

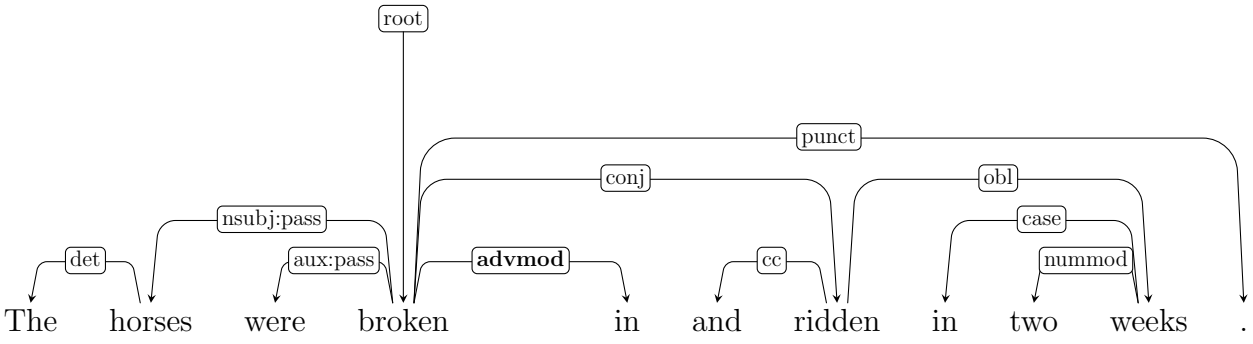
Finally, I performed quantitative evaluation of the parsers by running the MaltEval script[4], comparing the parsers' outputs to the gold standards. The results of this process are discussed in **??**.

The scripts, parses, evaluation results and gold standards can be found in the directory associated with this project[5].

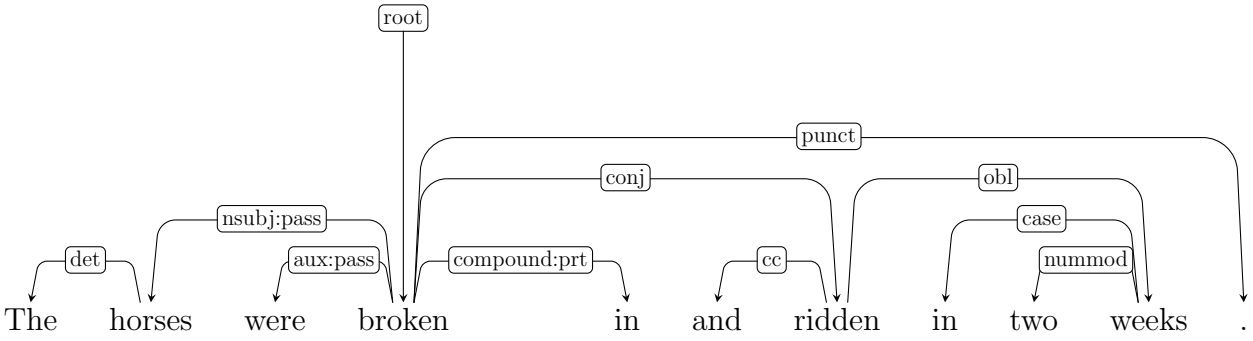
[2] `https://stanfordnlp.github.io/CoreNLP/`
[3] `https://github.com/jonorthwash/ud-annotatrix`
[4] `http://www.maltparser.org/malteval.html`
[5] `https://github.com/codebyzeb/l95`

4 Qualitative Analysis

In this section, I investigate the failures of the parsers when operating a set of 25 challenging sentences and suggest possible causes for these failures. An example of the parses for one of the shorter sentences in the set can be seen in **??**. This example leads to my first observation: that for the majority of dependencies, the parsers succeed. In most cases, the parsers correctly attach determiners, adjectives, adverbs and other local dependencies. The **obj** object dependency and the **nsubj** subject dependency of predicates also tend to be correctly assigned. In most cases, the parsers are able to handle these particular local and global dependencies for common constructions.



(a) Shift Reduce Parser.



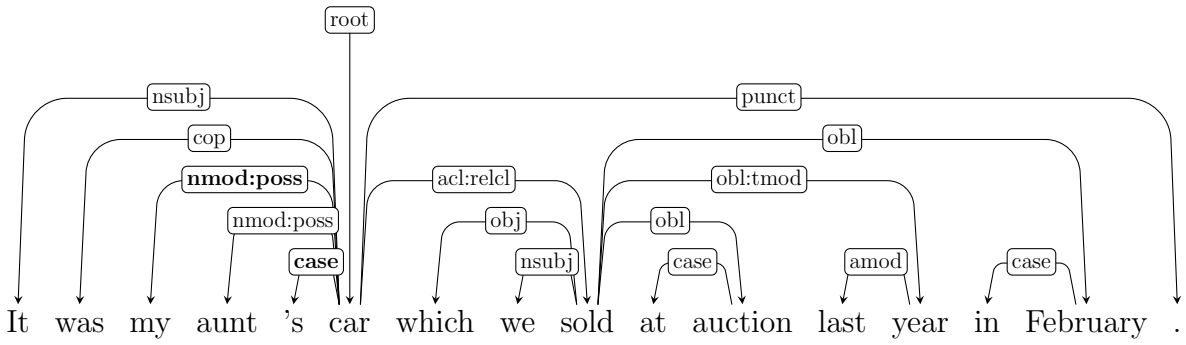
(b) Unlexicalised Parser, Neural Parser and Gold Standard.

Figure 4: Parses and Gold Standard for Sentence 3 with incorrect dependencies in **bold**.

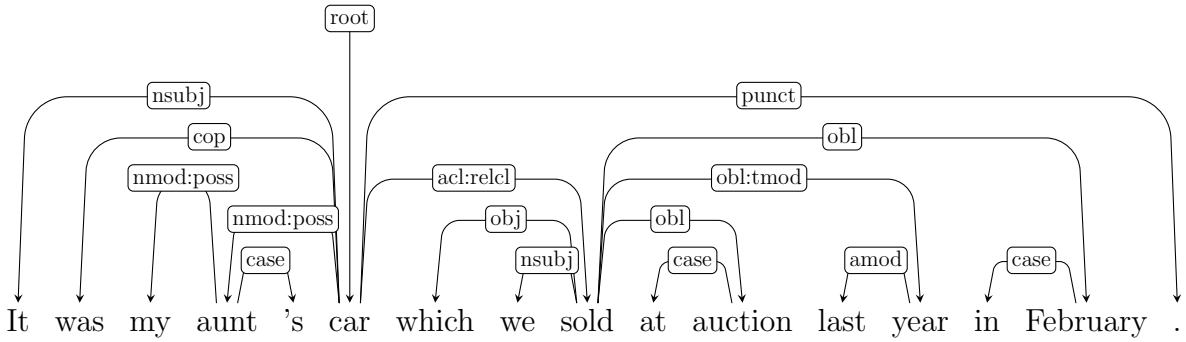
The failure cases discussed below are due to more complicated issues, including long-range dependencies, rare constructions and pre-processing difficulties. Many of these cases may not have appeared in the training data for the parsers, possibly explaining these failures, although this is not the case for all issues (such as punctuation). I find that the parsers struggle significantly with these constructions but that generally the Neural parser performs best.

4.1 Possessives

An interesting failure, demonstrated by just the Shift Reduce parser, is the failure to attach the possessive clitic to the correct nominal. One example of this is the parse of Sentence 7, seen in **??**. The parser incorrectly connects the **case** and **nmod:poss** dependencies to the wrong heads for "'s" and "my". It is possible that the greedy approach taken by the parser favours completing an NP over chaining multiple possessives, but this error also occurs when using the beam search model.



(a) Shift Reduce Parser.

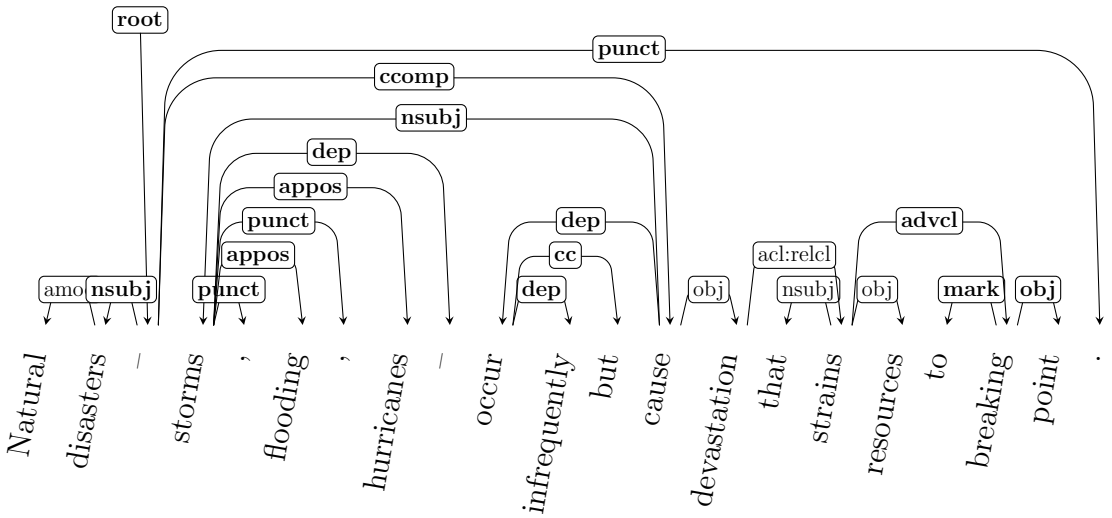


(b) Gold Standard.

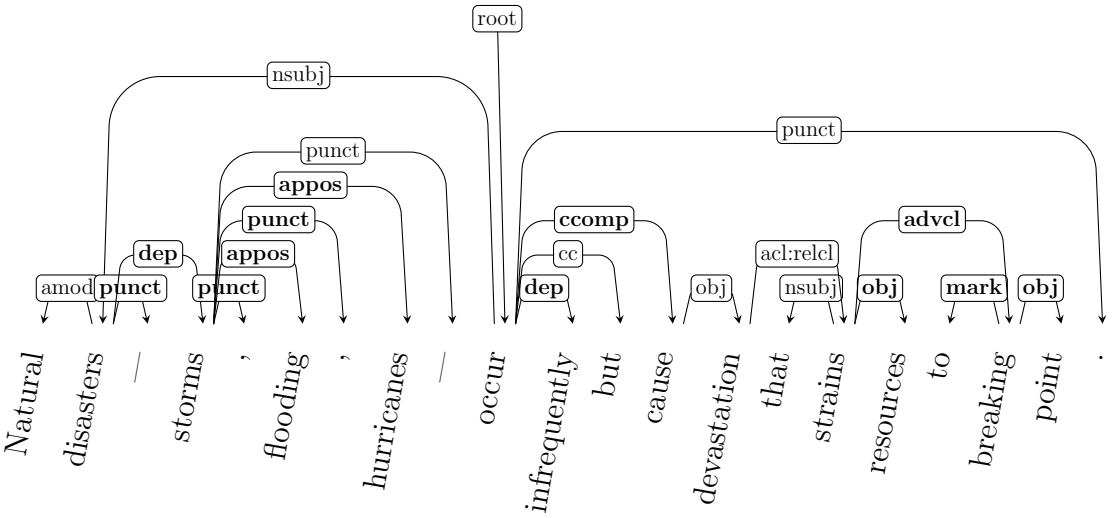
Figure 5: Parses and Gold Standard for Sentence 7 with incorrect dependencies in **bold**.

4.2 Pre-processing

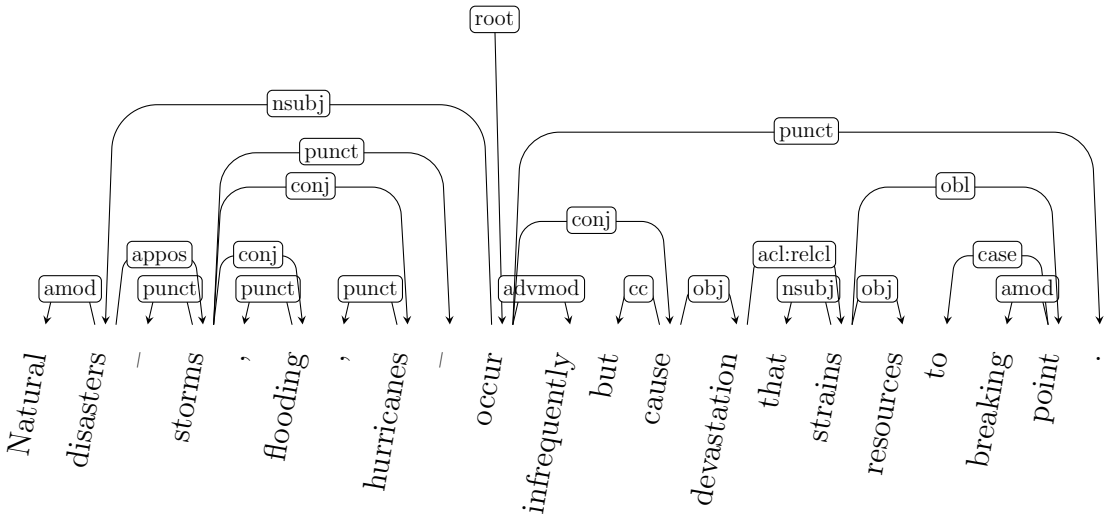
A consequence of the pre-processing method described in **??** is that some of the sentences were slightly altered, causing significant parsing failures. One example is Sentence 10, where the em-dash in the original sentence was converted to an en-dash. The parse of this sentence, seen in **??**, shows the Shift Reduce parser getting almost every dependency wrong as a result of this error. The part-of-speech tagger assigns the SYM tag to these dashes, rather than a punctuation tag. This tagging error explains some of the mistakes, the most significant of which is the fact that the Shift Reduce parser assigns the **root** dependency to one of the en-dashes. With the root of the tree incorrect, it it not surprising that so many other dependencies are assigned incorrectly. The Neural and Unlexicalised parsers also struggle, but not as severely (both get the root of the sentence correctly).



(a) Shift Reduce Parser, before correcting the em-dash.



(b) Shift Reduce Parser, after correcting the em-dash.

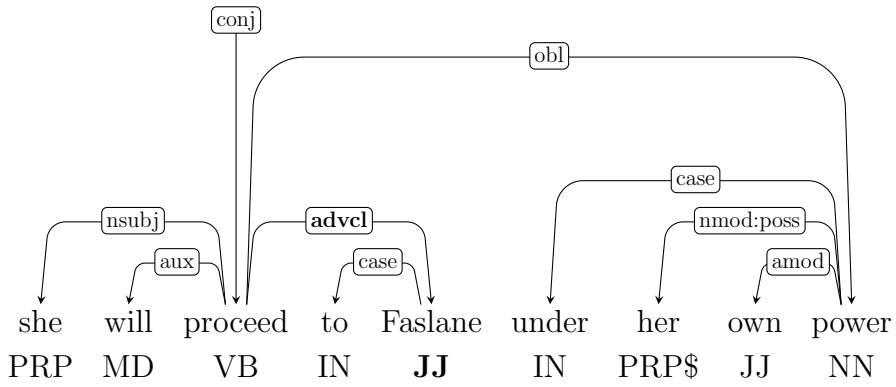


(c) Gold Standard.

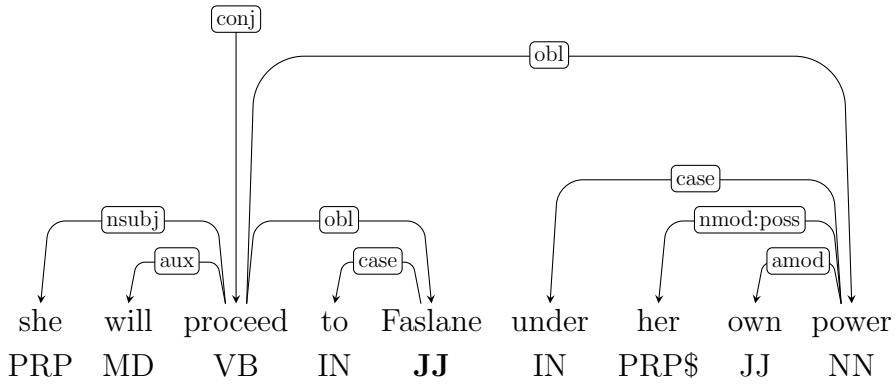
Figure 6: Parses and Gold Standard for Sentence 10 with incorrect dependencies in **bold**.

Correcting the pre-processed text to use the em-dash instead does fix some of these structural issues, as seen in **??**, but there are still many mistakes. Parsing a sentence with grammatical errors will

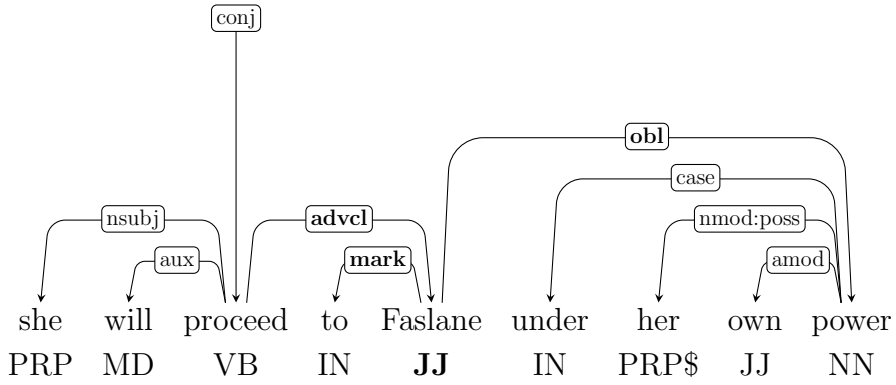
inevitably cause parsing failures but this raises questions as to whether these parsers should be more robust to this kind of 'noise'. If these parsers were being used to parse tweets, for examples, these kind of errors would be very common and it would be desirable for the parsers to be able to handle them. As such, I do not correct the input sentence for the quantitative evaluation done in **??**, I use the en-dash for both the input sentence and the gold standard.



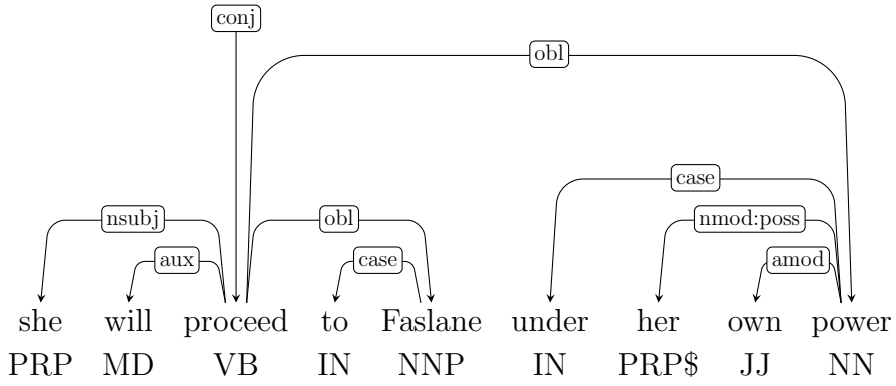
(a) Unlexicalised Parser.



(b) Neural Parser.



(c) Shift Reduce Parser.



(d) Gold Standard.

Figure 7: Parses and Gold Standard for a fragment of Sentence 22 with incorrect dependencies and part-of-speech tags in **bold**.

Part-of-speech tagging can be also be considered a pre-processing step as it operates separately and prior to parsing when run as part of the Stanford CoreNLP pipeline. As discussed in **??**, all parsers use the same part-of-speech information when processing the sentences. Although part-of-speech tagging is generally considered a "solved task", with modern systems achieving accuracies of 97%, mistakes are still made and this can cause issues in parsing.

In **??**, I show an example. The proper noun "Faslane" is incorrectly tagged with the JJ (adjective) tag. The parsers have likely not seen this unknown word in the training data. Hence, we would expect the parsers to mostly use this tag when assigning dependencies. Both the Unlexicalised parser and the Shift Reduce parser assign the **ADJP** (adjective phrase) label to the phrase consisting of the word "Faslane". The Unlexicalised parser does recognise "to Faslane" as a prepositional phrase, assigning PP in the phrase structure tree, but the **advcl** dependency is still incorrectly assigned, likely due to the dependency conversion script that runs over the constituency tree. The Shift Reduce parser creates an SBAR clause out of "to Faslane under her own power", resulting in the incorrect **advcl** and **mark** dependencies. There are no common syntactic constructions where an adjective follows "to" so it is understandable that the parsers get confused by this. It may be that the greedy algorithm used by the Shift Reduce parser prefers forming incorrect clauses to incorrect prepositional phrases whereas the chart parsing algorithm used by the unlexicalised parser is able to find that the prepositional phrase is more likely.

It is interesting that the Neural parser, which does not create a phrase structure tree, is able to get the correct parse here. It may be that given the other dependencies assigned and the part-of-speech tag for "to" (features considered by the neural network), it finds that **obl** is the most likely dependency to assign here. Perhaps by avoiding the creation of an Adjective Phrase, or any phrase structure, the parser has actually avoided the incorrect parse caused by the tagging error.

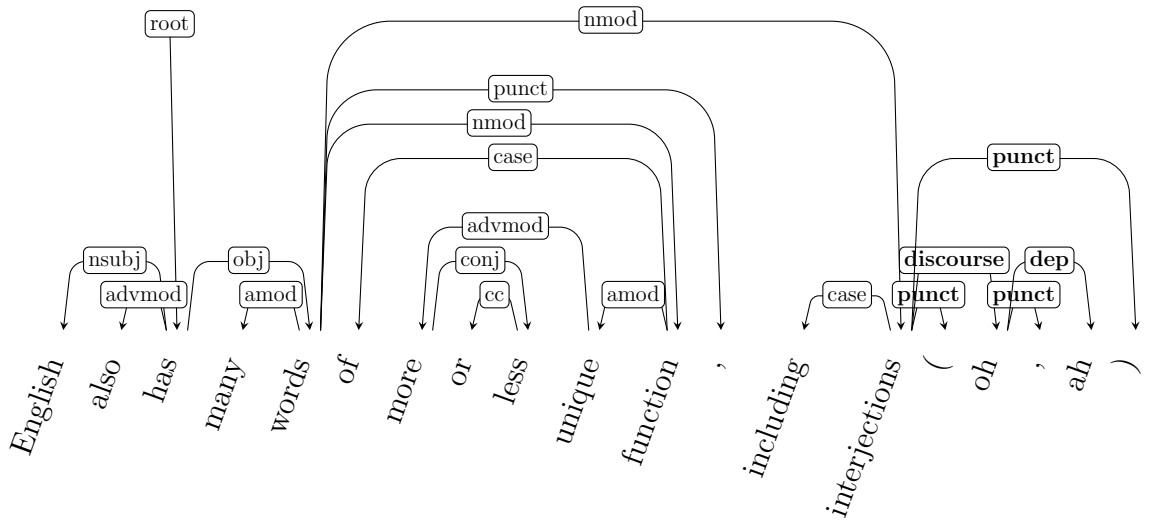
4.3 Use/Mention Distinction

Words can either be *used* or *mentioned*. Mentioning a word is to use it to refer to the word itself, an example of *metalanguage*. The *use-mention distinction* is the problem of distinguishing whether a word (or any element of language) is being used or mentioned, in both spoken and written language. In spoken language, there is evidence that we make significant use of the use-mention distinction (**?**) for a variety of metalanguage functions such as explaining meaning, introducing new words and reporting speech (**?**). In written language, we often indicate mentioned words using quotation marks or italics, the later being the case for Sentence 17 as seen in **??**.

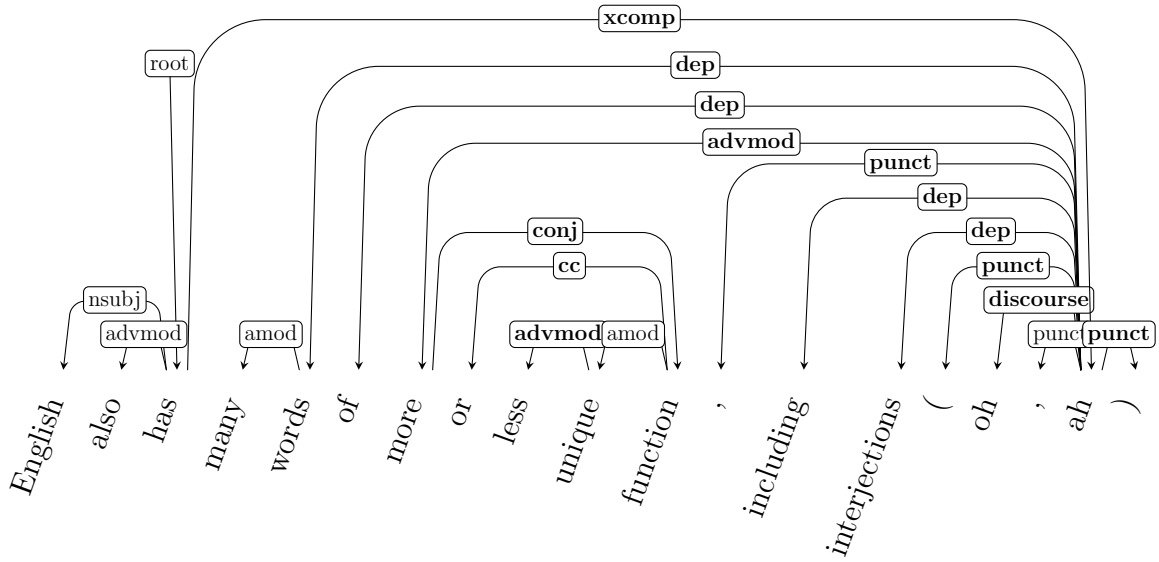
> The Penn Treebank tagset was culled from the original 87-tag tagset for the Brown Corpus. For example the original Brown and C5 tagsets include a separate tag for each of the different forms of the verbs *do* (e.g. C5 tag VDD for *did* and VDG tag for *doing*), *be* and *have*.

Figure 8: Original Formatting for Sentence 17

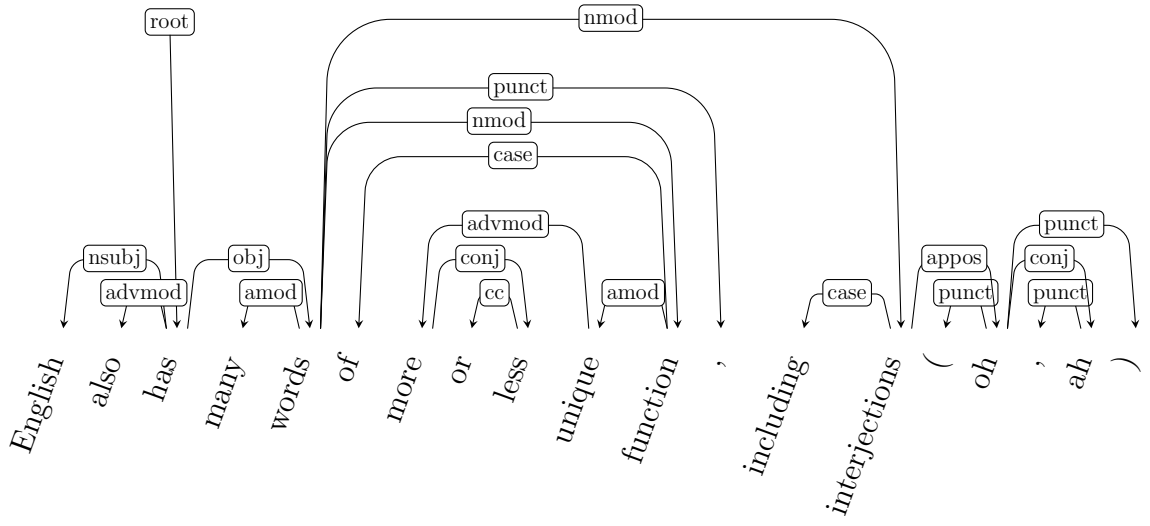
Although initial work has investigated building systems that make this distinction (**?**), the Stanford parsers do not model this directly and the part-of-speech tagger does not explicitly tag mentioned words as nouns. Thus, even if we had not lost the italics in the pre-processing step, the parsers would still not be able to understand that the italicised words in Sentence 17 and the parenthesised words in Sentence 15 are being *mentioned* and not *used*. This causes all three parsers to struggle with these two sentences.



(a) Neural Parser.



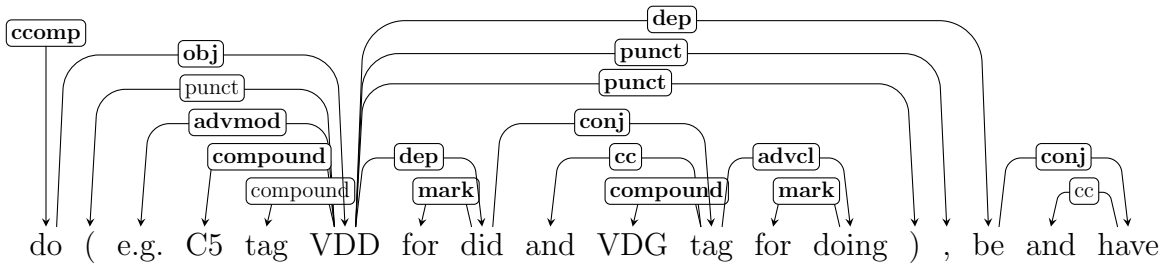
(b) Shift Reduce Parser.



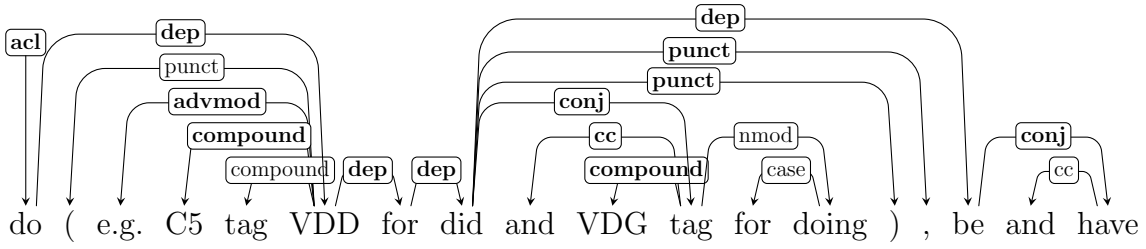
(c) Gold Standard.

Figure 9: Parses and Gold Standard for the a fragment of Sentence 15 with incorrect dependencies in **bold**.

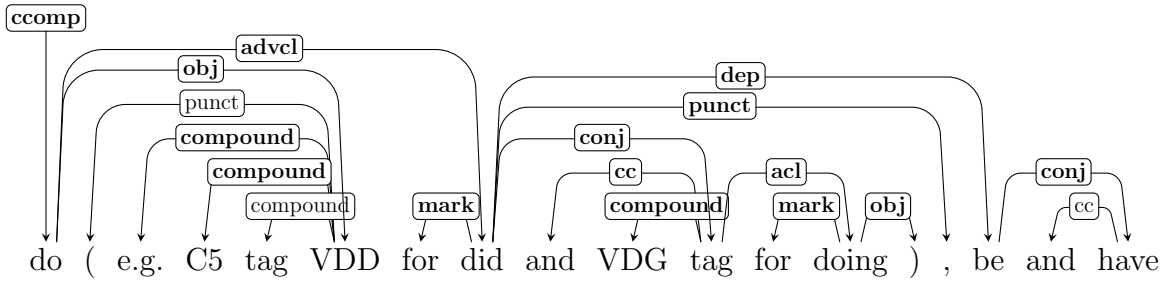
The parses for the first nineteen tokens of Sentence 15 are shown in **??**. Similarly to Sentence 10, the Shift Reduce parser significantly struggles with the structure of this sentence. The parser incorrectly considers "ah" to be the root of a major subtree in this parse and this again results in a series of other incorrect assignments. Four **dep** dependencies are assigned here. The **dep** label is used to capture parser uncertainty, indicating that the parser is unable to identify the correct structure of the sentence. The Neural parser gets the overall structure correct but is also confused by the mentioned words, using the **discourse** relation to connect "oh". The Unlexicalised parser makes a similar error.



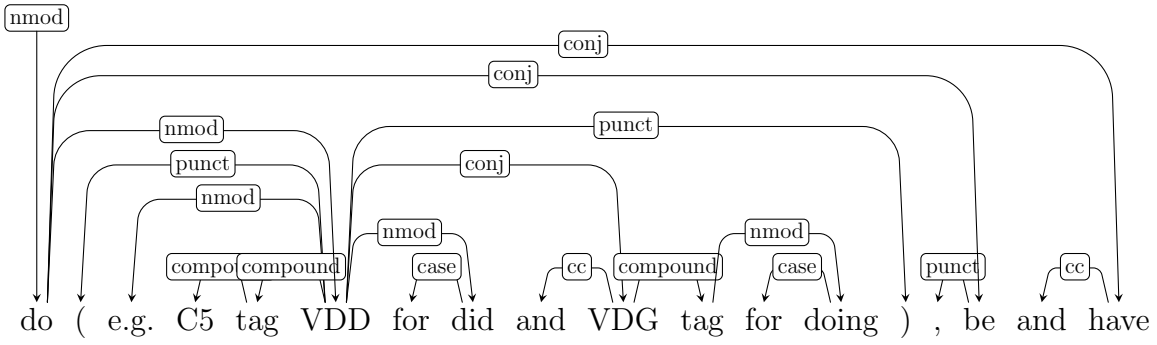
(a) Unlexicalised Parser.



(b) Neural Parser.



(c) Shift Reduce Parser.



(d) Gold Standard.

Figure 10: Parses and Gold Standard for a fragment of Sentence 17 with incorrect dependencies in **bold**.

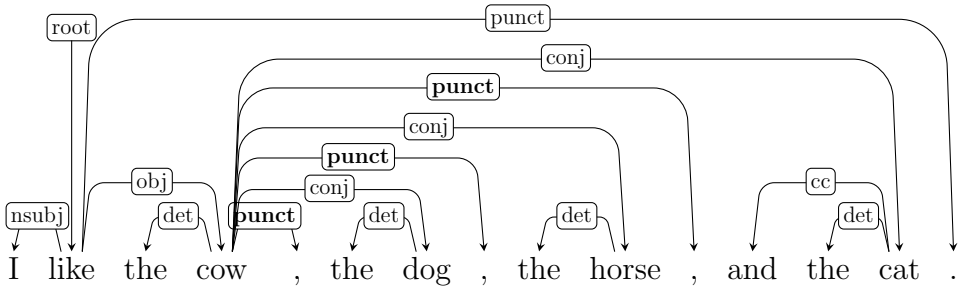
The parses for a fragment of Sentence 17 are shown in **??**. All three parsers significantly struggle due to the use/mention distinction problem, all getting almost every dependency wrong. The correct parse is to create a conjunction of the words "do", "be" and "have". The parsers are able to find a

conjunction between "be" and "have" but all miss the conjunction with "do". We also see all three parsers frequently using the **dep** relation, particularly the Neural parser which uses it four times in this short fragment. The only successful tagging is done by the Neural parser for the prepositional phrase "for doing", correctly assigning the **nmod** and **case** dependencies. This may be similar to the success of the Neural parser for the part-of-speech error discussed in **??**: by not creating phrase structure trees, the Neural parser is able to avoid incorrect parses when the Unlexicalised and Shift Reduce can not.

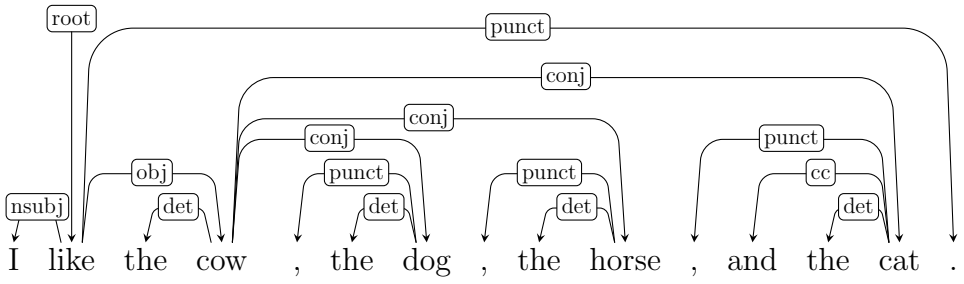
The strangest outputted dependency is the **obj** label outputted by the Shift Reduce parser between "doing" and the right bracket, seen in **??**. Given the greedy algorithm, this suggests that the perceptron's weights are so strongly weighted towards providing an object to the verb "doing" that it is able to overcome whatever weight exists to ensure that punctuation is only ever assigned the **punct** dependency. This example shows that the use/mention distinction can cause rather severe issues in parsing, particularly when the words being mentioned are as common as those in Sentence 17.

4.4 Punctuation

Across the 25 sentences, all three parses consistently fail to correctly parse punctuation. One of the rules for assigning the **punct** dependency that seems to be consistently disregarded is that paired punctuation marks should be assigned to the same word, and this word should be the head of the phrase enclosed in this pair. We see this in the parses for Sentence 10 (**??**), where the two em-dashes are not attached to "storms", and in the parses for Sentence 15 and Sentence 17 (**????**), where the brackets are not attached to the head of the enclosed phrases. This behaviour can also be seen in the example shown in **??**.



(a) Unlexicalised Parser, Neural Parser and Shift Reduce Parser.



(b) Gold Standard.

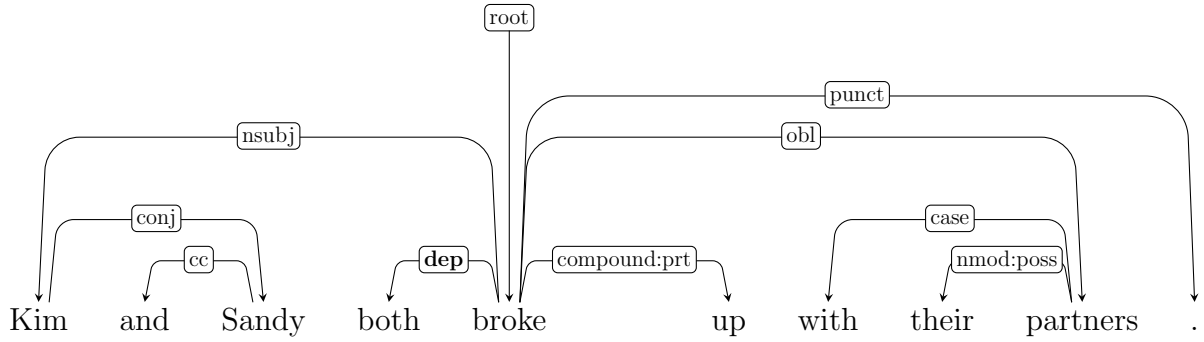
Figure 11: Parses and Gold Standard for a simple conjuction with incorrect dependencies in **bold**.

Another rule that is consistently disregarded is that commas in a conjunction should attach to the head of the following conjuncts. Instead, these parsers seem to incorrectly favour attaching the **punct** dependency to the head of the *previous* conjunct. This is also seen in all three parses for both Sentence 10 and Sentence 15. It could be argued that the incorrect attachment of commas for

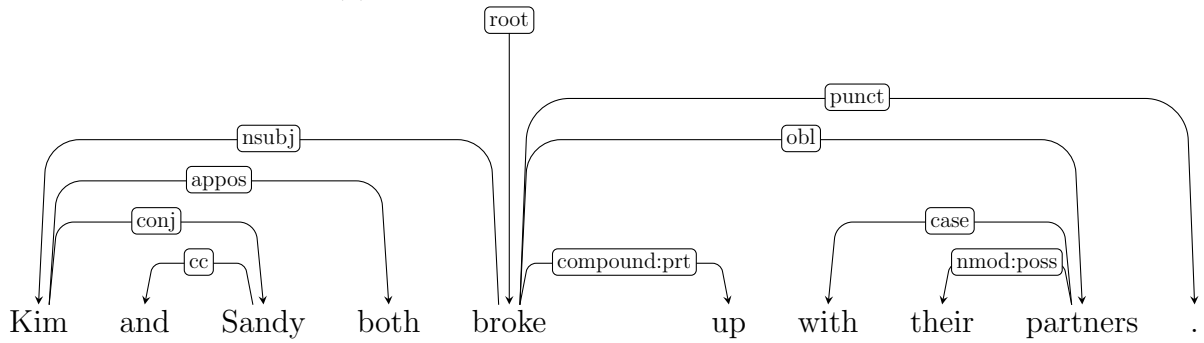
these two sentences is due to the fact that the parsers fail to recognise that conjunction occurs at all, as discussed in **??**, however the parsers also fail when they do detect conjunction, as seen in **??**. This is not an edge case of parsing and so should be treated as a serious parser failure. This issue may be due to mislabelled training data, or perhaps an algorithmic failure to model differences in attachment for punctuation in training.

4.5 Apposition

Apposition is a construction that involves two elements being placed side by side such that one provides an alternative identification of the other. Constructions of this sort occur frequently in the set of 25 sentences and the parsers all struggle with it.



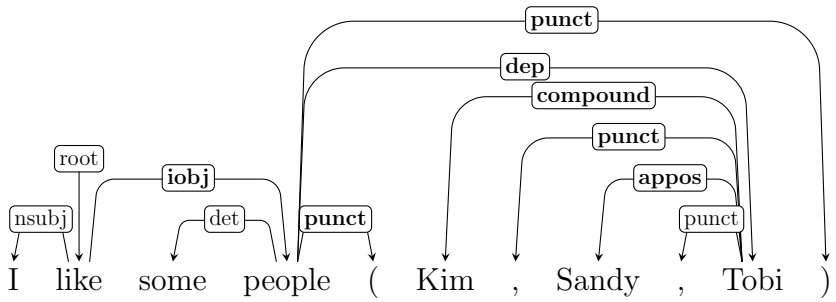
(a) Unlexicalised and Shift Reduce Parser.



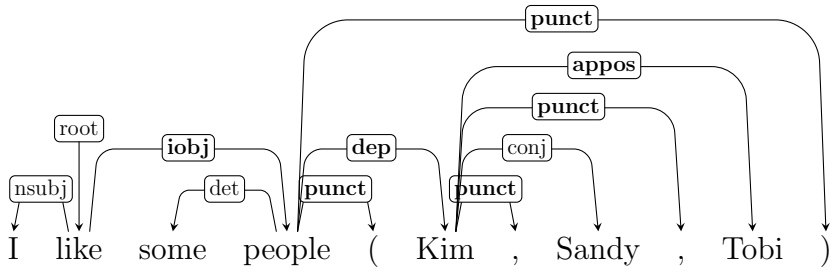
(b) Neural Parse and Gold Standard.

Figure 12: Parses and Gold Standard for Sentence 4 with incorrect dependencies in **bold**.

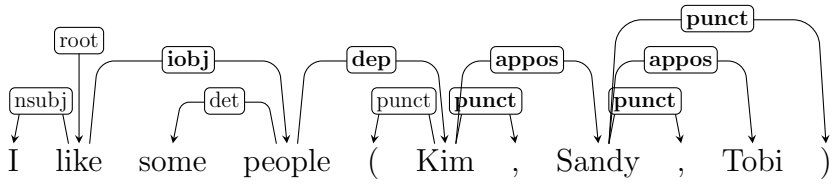
One example can be seen in **??**. The parsers struggle with the phrase "Kim and Sandy both". It is an unusual use of a pre-determiner placed after a nominal, but it is grammatical. If "both" is moved to the start of the sentence, all parsers correctly create a **cc:preconj** dependency with it and "Kim". In this case, however, "both" seems to be in apposition with "Kim and Sandy" and so an **appos** relation should be assigned, not the **dep** relation assigned by the Unlexicalised and Shift Reduce parser.



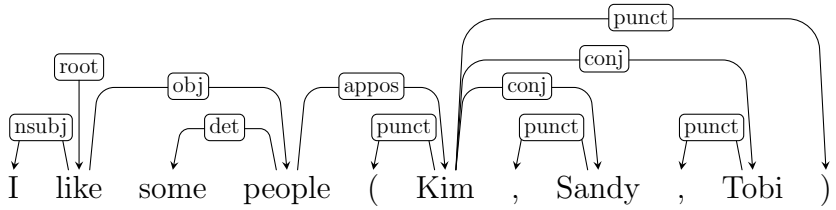
(a) Unlexicalised Parser.



(b) Neural Parser.



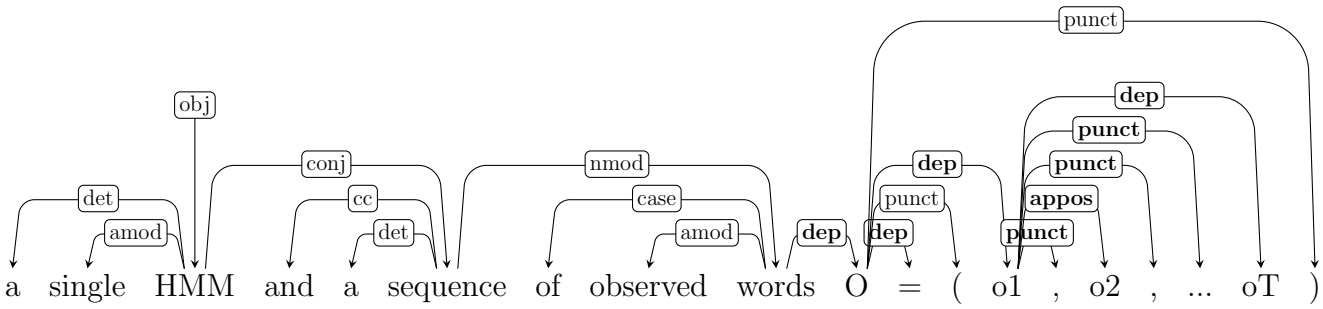
(c) Shift Reduce Parser.



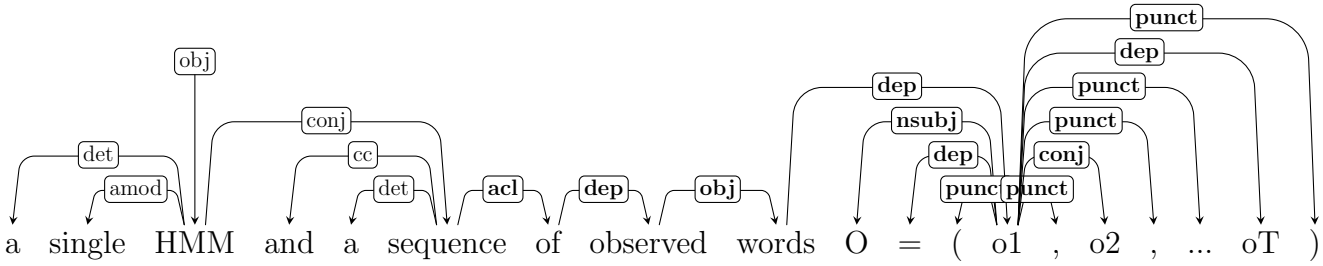
(d) Gold Standard.

Figure 13: Parses and Gold Standard for a sentence containing apposition and conjunction without coordination with incorrect dependencies in **bold**.

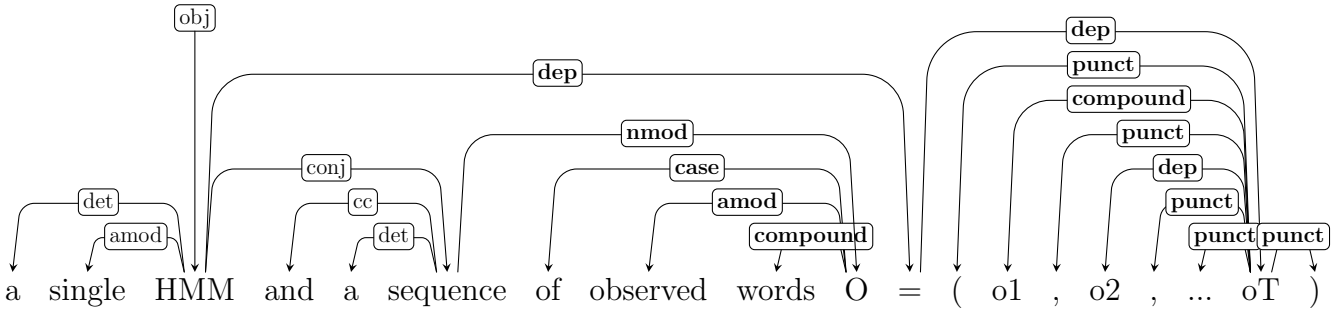
According to the manual, the **appos** dependency should also be used for *parenthesised examples*. In **??** we see that the Neural and Shift Reduce parser fail to use the **appos** dependency for linking the parenthesised example in Sentence 15 "(oh, ah)" to "interjections". The Unlexicalised parser also fails to do this, using the **dep** relation instead. Even for simple sentences, like the one shown in **??**, the parsers fail to correctly assign the **appos** relation to the parenthesised examples, using **dep** instead. Interestingly, they do use the **appos** relation *within* the bracketed expression. This is due to the fact that conjunction and apposition can both be delimited by commas; a phrase such as "I like Kim, the butcher, and the dog" is ambiguous as to whether "Kim" and "the butcher" refer to the same entity.



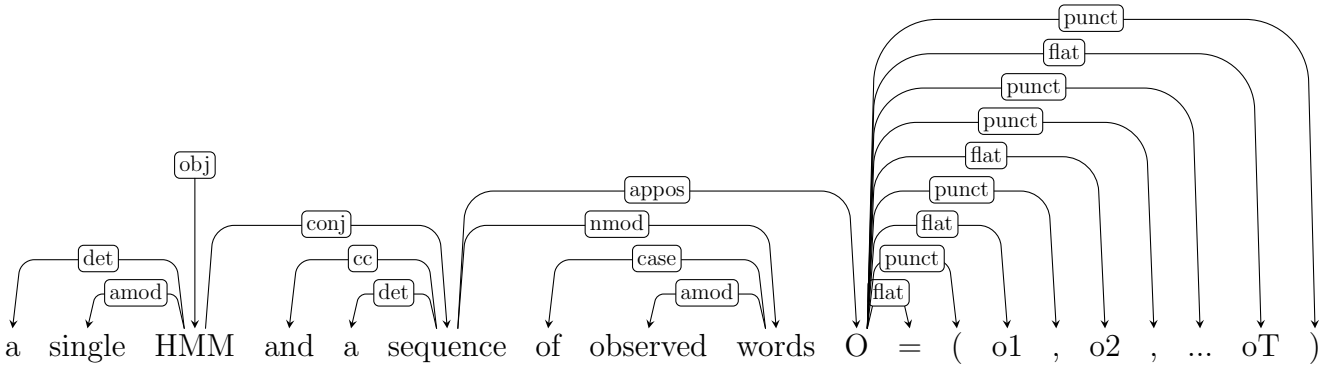
(a) Unlexicalised Parser.



(b) Neural Parser.



(c) Shift Reduce Parser.



(d) Gold Standard.

Figure 14: Parses and Gold Standard for a fragment of Sentence 18 with incorrect dependencies in **bold**.

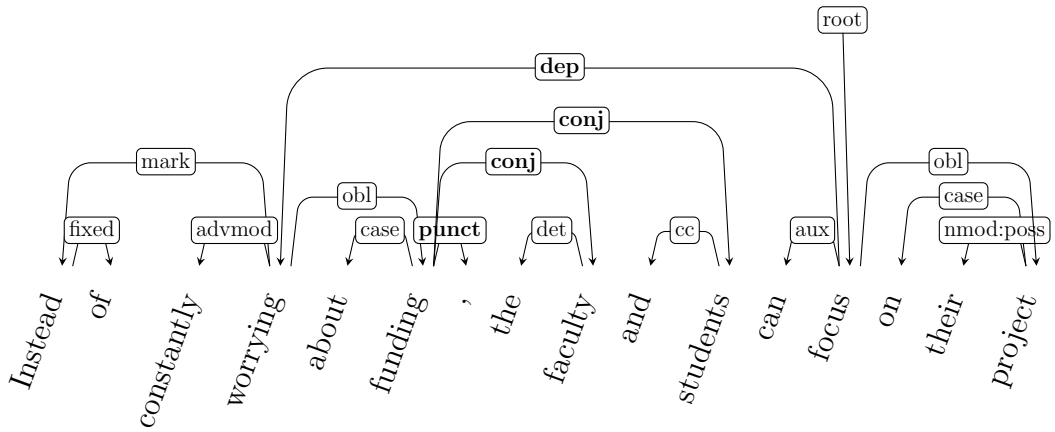
Another edge case of apposition can be seen in **??**, which presents the parses for a fragment of Sentence 18. Here, the equation is in apposition with the noun phrase "a sequence of observed words" and so an **appos** dependency should connect the two. None of the parsers use any **appos** labels for this fragment, likely due to the difficulty in parsing the equation, which is a difficult task as mathematical expressions do not follow the same grammatical rules as English. For the gold standard parse, I use a combination of **flat** and **punct** relations all linked to the first symbol of the equation. I found that this labelling was the best way to represent the entire equation as a single multi-word expression.

Although it is an edge-case, I believe that this is what we would like parsers to be capable of: detecting when a sequence of tokens may be part of a different grammatical system (such as mathematics) and preventing the symbols used in this sequence from interfering with the parse for the rest of the sentence. The Unlexicalised parser achieves this, getting every dependency outside of the subtree containing the mathematical expression correct. The Neural parser also constrains the expression to a subtree, but fails to parse "of observed words", possibly as a result of interference with this subtree. The Shift Reduce parser does not constrain the expression and this results in numerous other dependencies being incorrect.

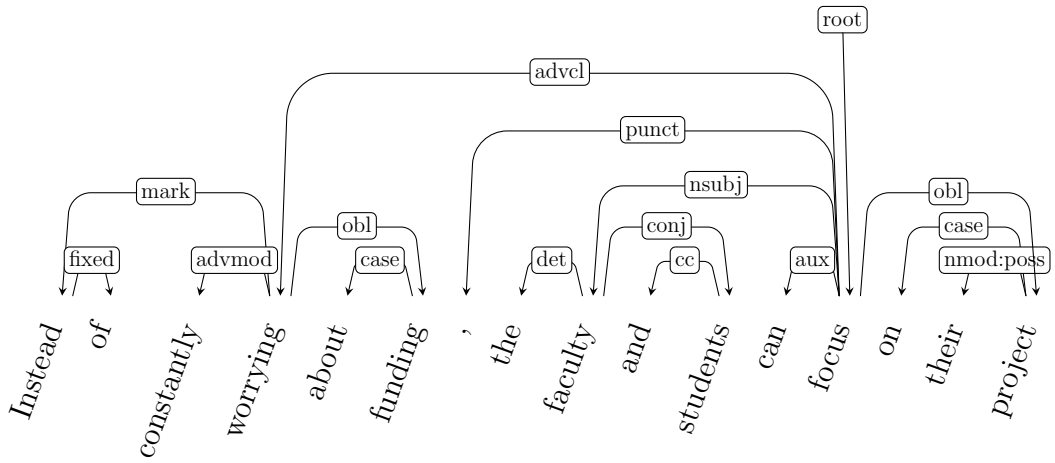
4.6 Conjunction

A frequent error exhibited by all parsers is to do with conjunction. When conjuncts are clearly delimited with commas and coordinating conjunctions, all parsers succeed in assigning the **conj** relation as seen in **??**. For many of the sentences in this dataset, however, the parsers fail to detect conjunction.

One example is in Sentence 15 (see **??**) where conjunction appears for each parenthesised example. It seems that due to the lack of coordinating conjunction, the parsers fail to see these pairs as conjunction, using a variety of other dependencies instead. We also see this behaviour occur for simpler examples, as in **??**, where the a combination of **appos** and **conj** dependencies are used rather than correctly assigning two **conj** labels to link the three conjuncts. We also see **appos** being assigned rather than **conj** when parsing Sentence 10, seen in **??**.



(a) Unlexicalised Parser.



(b) Gold Standard.

Figure 15: Parses and Gold Standard for a fragment of Sentence 24 with incorrect dependencies in **bold**.

Another failure associated with conjunction is shown in **??**. The Unlexicalised parser creates a conjunction out of "funding, the faculty and students", which is a valid conjunction but renders the remaining sentence ungrammatical. Given that the parser uses a chart parsing algorithm, which should find an optimal parser, it is surprising that it does not find that the smaller conjunction "the faculty and students" to be more likely.

The parsers also fail to find the conjunction in Sentence 17 (**??**) but this is mostly due to the use/mention distinction problem as discussed in **??**.

5 Quantitative Evaluation

In order to verify the observations made in **??**, I evaluated the parsers using the MaltEval script to compare the parses to the gold standards. I used two metrics; the `BothRight` metric which marks a dependency as correct if both the label and head are the same as in the gold standard and the `LabelRight` metric which marks a dependency as correct if just the label is correct. This distinction is important since by default, the MaltEval script uses the `LabelRight` metric when grouping results by dependency label. Comparing the two evaluation metrics can provide insight into cases where the label is correctly assigned but the dependency is connected to the wrong head.

	Neural Parser			Unlexicalised Parser			Shift Reduce Parser		
Dependency	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
acl	33.3	33.3	33.3	-	0.0	-	33.3	33.3	33.3
acl:relcl	92.9	86.7	89.7	100.0	73.3	84.6	81.2	86.7	83.9
advcl	80.0	57.1	66.6	18.2	28.6	22.2	50.0	42.9	46.2
advmod	89.5	89.5	89.5	89.5	89.5	89.5	87.2	89.5	88.3
amod	97.7	89.6	93.5	100.0	87.5	93.3	93.3	87.5	90.3
appos	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
aux	88.2	100.0	93.7	100.0	100.0	100.0	100.0	100.0	100.0
aux:pass	100.0	100.0	100.0	100.0	100.0	100.0	83.3	83.3	83.3
case	95.5	96.9	96.2	96.9	95.4	96.1	98.4	93.8	96.0
cc	93.3	100.0	96.5	96.6	100.0	98.3	100.0	96.4	98.2
ccomp	80.0	66.7	72.7	55.6	83.3	66.7	38.5	83.3	52.7
compound	83.3	92.6	87.7	78.6	81.5	80.0	68.8	81.5	74.6
compound:prt	100.0	80.0	88.9	100.0	60.0	75.0	100.0	80.0	88.9
conj	84.8	77.8	81.1	79.3	63.9	70.8	76.9	55.6	64.5
cop	90.9	100.0	95.2	90.0	90.0	90.0	80.0	80.0	80.0
csubj	0.0	0.0	0.0	50.0	100.0	66.7	100.0	100.0	100.0
dep	0.0	-	-	0.0	-	-	0.0	-	-
det	98.0	96.1	97.0	98.1	100.0	99.0	98.0	98.0	98.0
det:predet	100.0	100.0	100.0	100.0	100.0	100.0	50.0	100.0	66.7
discourse	0.0	-	-	0.0	-	-	0.0	-	-
expl	100.0	100.0	100.0	50.0	50.0	50.0	66.7	100.0	80.0
fixed	83.3	83.3	83.3	100.0	83.3	90.9	100.0	83.3	90.9
flat	-	0.0	-	-	0.0	-	-	0.0	-
iobj	66.7	100.0	80.0	66.7	100.0	80.0	66.7	100.0	80.0
mark	100.0	78.6	88.0	76.5	92.9	83.9	78.6	78.6	78.6
nmod	83.3	78.1	80.6	79.3	71.9	75.4	85.2	71.9	78.0
nmod:poss	100.0	100.0	100.0	100.0	100.0	100.0	100.0	91.7	95.7
nsubj	94.3	90.9	92.6	88.9	87.3	88.1	83.1	89.1	86.0
nsubj:pass	100.0	100.0	100.0	100.0	100.0	100.0	83.3	83.3	83.3
nummod	100.0	88.9	94.1	100.0	88.9	94.1	87.5	77.8	82.4
obj	82.4	80.0	81.2	87.9	82.9	85.3	83.8	88.6	86.1
obl	81.8	84.4	83.1	84.4	84.4	84.4	86.7	81.2	83.9
obl:npmod	100.0	100.0	100.0	66.7	100.0	80.0	100.0	100.0	100.0
obl:tmod	66.7	100.0	80.0	66.7	100.0	80.0	100.0	100.0	100.0
parataxis	60.0	50.0	54.5	40.0	33.3	36.3	50.0	16.7	25.0
punct	100.0	97.9	98.9	100.0	97.9	98.9	100.0	95.8	97.9
root	82.1	82.1	82.1	89.3	89.3	89.3	85.7	85.7	85.7
xcomp	80.0	57.1	66.6	100.0	85.7	92.3	75.0	85.7	80.0
macroaverage	**78.1**	**78.8**	**81.3**	76.4	77.8	80.6	74.9	78.4	78.8
microaverage	**87.2**	**89.7**	**90.2**	85.5	87.5	88.0	84.3	87.0	86.9

Table 1: Accuracy of the parsers using the `LabelRight` MaltEval metric with highest microaverage and macroaverage scores in **bold**.

I used the MaltEval script's `GroupBy` parameter to calculate the the precision, recall and F_1 scores for each dependency. These scores provides more insight than a single overall accuracy score as some labels occur more frequently than others. The results for the `BothRight` and `LabelRight` metrics can be seen in **??** and **??** respectively.

	Neural Parser			Unlexicalised Parser			Shift Reduce Parser		
Dependency	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
acl	33.3	33.3	33.3	-	0.0	-	33.3	33.3	33.3
acl:relcl	85.7	80.0	82.8	90.9	66.7	76.9	56.2	60.0	58.0
advcl	20.0	14.3	16.7	9.1	14.3	11.1	33.3	28.6	30.8
advmod	78.9	78.9	78.9	81.6	81.6	81.6	76.9	78.9	77.9
amod	93.2	85.4	89.1	95.2	83.3	88.9	77.8	72.9	75.3
appos	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
aux	82.4	93.3	87.5	93.3	93.3	93.3	93.3	93.3	93.3
aux:pass	100.0	100.0	100.0	100.0	100.0	100.0	83.3	83.3	83.3
case	92.4	93.8	93.1	93.8	92.3	93.0	93.5	89.2	91.3
cc	83.3	89.3	86.2	79.3	82.1	80.7	77.8	75.0	76.4
ccomp	80.0	66.7	72.7	44.4	66.7	53.3	38.5	83.3	52.7
compound	76.7	85.2	80.7	67.9	70.4	69.1	62.5	74.1	67.8
compound:prt	100.0	80.0	88.9	100.0	60.0	75.0	100.0	80.0	88.9
conj	72.7	66.7	69.6	69.0	55.6	61.6	65.4	47.2	54.8
cop	72.7	80.0	76.2	70.0	70.0	70.0	80.0	80.0	80.0
csubj	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	100.0
dep	0.0	-	-	0.0	-	-	0.0	-	-
det	96.0	94.1	95.0	94.2	96.1	95.1	88.2	88.2	88.2
det:predet	100.0	100.0	100.0	100.0	100.0	100.0	50.0	100.0	66.7
discourse	0.0	-	-	0.0	-	-	0.0	-	-
expl	50.0	50.0	50.0	0.0	0.0	0.0	33.3	50.0	40.0
fixed	83.3	83.3	83.3	100.0	83.3	90.9	100.0	83.3	90.9
flat	-	0.0	-	-	0.0	-	-	0.0	-
iobj	66.7	100.0	80.0	66.7	100.0	80.0	66.7	100.0	80.0
mark	81.8	64.3	72.0	52.9	64.3	58.0	71.4	71.4	71.4
nmod	80.0	75.0	77.4	79.3	71.9	75.4	81.5	68.8	74.6
nmod:poss	100.0	100.0	100.0	100.0	100.0	100.0	100.0	91.7	95.7
nsubj	84.9	81.8	83.3	83.3	81.8	82.5	78.0	83.6	80.7
nsubj:pass	100.0	100.0	100.0	100.0	100.0	100.0	83.3	83.3	83.3
nummod	100.0	88.9	94.1	100.0	88.9	94.1	87.5	77.8	82.4
obj	79.4	77.1	78.2	87.9	82.9	85.3	78.4	82.9	80.6
obl	75.8	78.1	76.9	81.2	81.2	81.2	76.7	71.9	74.2
obl:npmod	100.0	100.0	100.0	66.7	100.0	80.0	100.0	100.0	100.0
obl:tmod	66.7	100.0	80.0	33.3	50.0	40.0	100.0	100.0	100.0
parataxis	20.0	16.7	18.2	20.0	16.7	18.2	50.0	16.7	25.0
punct	48.9	47.9	48.4	57.4	56.2	56.8	48.9	46.9	47.9
root	82.1	82.1	82.1	89.3	89.3	89.3	85.7	85.7	85.7
xcomp	80.0	57.1	66.6	100.0	85.7	92.3	75.0	85.7	80.0
macroaverage	**69.4**	**70.6**	**72.6**	66.9	66.2	69.9	68.3	71.3	71.8
microaverage	**75.1**	**77.5**	**77.9**	74.8	76.0	76.7	71.3	73.9	73.6

Table 2: Accuracy of the parsers using the `BothRight` MaltEval metric with highest microaverage and macroaverage scores in **bold**.

For each score type, I calculated the macroaverage and microaverage. The former is calculated by taking the average of each measure for each dependency. The latter is similar but weights each by label occurrence. These averages represent the performance of the parsers in slightly different ways, the first weights all dependencies equally and the latter is more representative of overall accuracy.

These averages confirm that the Neural parser indeed performs better on this set of difficult sentences, achieving higher microaverage and macroaverage scores than the other two parsers for every score type. All parsers achieve higher microaverage scores to macroaverage scores, suggesting that although they may struggle with correctly assigning particular dependencies, they get the most frequent dependencies correct. This result follows my initial observation that common local and global dependencies are often correctly assigned, further confirmed by the high scores for **advmod**, **amod**, **aux**, **cc**, **det**, **fixed**, **nsubj**, **nummod**, **obj**, and **obl**.

The lower scores confirm the parser failures discussed in **section:qual**. One example is that all parsers get a score of 0 for precision and recall for the **appos** relation. Despite being a rarer construction, this is still a noteworthy failure. The table also shows that the parsers struggle with assigning the **parataxis** dependency, another rare construction. The rarity and complexity of these constructions may explain why these parsers produce these error but also limits the impact of these failures on the overall accuracy scores reported for these parsers. This indicates the need to study the performance of parsers under difficult conditions and report scores for specific labels.

The scores for the **punct** dependency for the **LabelRight** metric seem to suggest that the parsers do not struggle with assigning this dependency, but the scores are very low for the **BothRight** metric. It means that the parsers are able to assign the correct label, but rarely connect the dependency to the correct head, as discussed in **??**. There is also a drop in scores between the two metrics for the **conj** dependency, indicating that although parsers are able to find conjuncts, they often connect them to the wrong head, though this is not as prominent as the drop for the **punct** dependency.

6 Discussion

The qualitative and quantitative analyses completed in this report demonstrate the two claims; that the Neural Parser is more accurate on difficult input and that there are certain constructions that all three parsers struggle with. It is important to note, however, that there are several limitations with my evaluation process. These findings should therefore be considered preliminary, with further work required to confirm these findings.

The first limitation is with the size of the dataset used to evaluate these parsers. Using only 25 sentences means that some dependencies only appeared once or not at all in the gold standards, such as **csubj** (1 occurrence) or **discourse** (0 occurrences). This results in skewed values in **????**, such as the scores of 0 and 100 for **csubj**, which can have a significant impact on the microaverage and macroaverage scores reported. The microaverage score in particular greatly depends on the distribution of dependency labels in the gold standards. These scores are still valid for cross-comparison, as I have done here, although a larger sample of "difficult" sentences with a detailed explanation of the distribution of gold standard labels would be required to be able to validate these findings.

Another limitation has to do with the construction of the gold standards. I constructed it alone, with my own limited linguistic knowledge, with only the Universal Dependencies manual for guidance. Although I did discuss my gold standard parses with another student, the gold standards should not be considered perfect, especially considering that these "difficult" sentences contain linguistic constructions with undocumented edge cases that required guesswork and linguistic intuition to label. Combined with the small sample size, any slight change in interpretation could have led to different results at the evaluation stage. Future work would need to systematically validate the gold standards, perhaps using labelling assistance software and majority voting.

A smaller limitation has to do with the calculation of the macroaverage scores. As some parsers did not achieve a precision score for some dependencies, these precision and F_1 scores for these dependencies were not included in the calculation of the microaverage and macroaverage, skewing

the average. One example is the precision score for **acl** for the Unlexicalised parser not being include in the average whereas the other two parsers included a score of 33.3, decreasing their averages in comparison. The effect is minimal and is another consequence of the small sample size but should be noted nonetheless.

7 Conclusion

I have described three of the Stanford parsers and explored their output on a set of 25 difficult sentences. I found that the Neural parser is more capable of parsing these difficult sentences and also that there are a number of constructions that all three systems struggle to parse. I confirmed these claims by examining the precision, recall and F_1 scores for each dependency and comparing the microaverage and macroaverage scores achieved by each parser. Finally, I noted that there are certain parts of this evaluation that would require further study, such as increasing the size of the dataset and systematically verifying the validity of the gold standards.

References

Anderson, M. L., Okamoto, Y., Josyula, D., and Perlis, D. (2002). The use-mention distinction and its importance to hci. In *Proceedings of the Sixth Workshop on the Semantics and Pragmatics of Dialog*, pages 21–28. Citeseer.

Bauer, J. (2014). Shift-reduce constituency parser. `https://nlp.stanford.edu/software/srparser.html`. Accessed: 2020-12-12.

Chen, D. and Manning, C. D. (2014). 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.

De Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–4592.

De Marneffe, M.-C., MacCartney, B., Manning, C. D., et al. (2006). Generating typed dependency parses from phrase structure parses. In *Lrec*, volume 6, pages 449–454.

Goldberg, Y. and Elhadad, M. (2011). Learning sparser perceptron models. Technical report, Tech. Rep.[Online]. Available: http://www. cs. bgu. ac. il/˜ yoavg/publications.

Goldberg, Y. and Nivre, J. (2012). A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976.

Hindle, D. and Rooth, M. (1993). Structural ambiguity and lexical relations. *Computational linguistics*, 19(1):103–120.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting of the association for computational linguistics*, pages 423–430.

Nivre, J., De Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., et al. (2016). Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Sagae, K. and Lavie, A. (2005). A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132.

Saka, P. (1998). Quotation and the use-mention distinction. *Mind*, 107(425):113–135.

Universal Dependencies contributors (2014). Universal dependencies v1 documentation. [Online; accessed 14-December-2020].

Wilson, S. (2010). Distinguishing use and mention in natural language. In *Proceedings of the NAACL HLT 2010 Student Research Workshop*, pages 29–33.

Zhang, Y. and Clark, S. (2009). Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171.

Zhu, M., Zhang, Y., Chen, W., Zhang, M., and Zhu, J. (2013). Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443.