A Comparison of Dependency Parsers

Hope Elizabeth McGovern
Cambridge Computer Laboratory
hem52@cam.ac.uk

Word Count: 4,995

Abstract

In this paper, we use grammatical relations to compare the accuracy of two off-the-shelf parsers, SpaCy and the Stanford parser, on a curated set of test sentences. We attempt to provide both a qualitative explanation of the types of errors made by each of the parsers as well as a quantitative report of their accuracy on a manually-created gold-label set of unbound dependencies. We compare and contrast the parsers for use in different situations.[1]

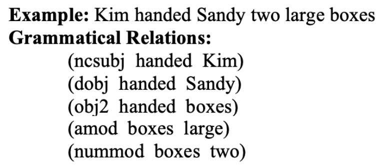
1 Introduction

Grammatical parsing is a foundational step in many natural language processing (NLP) tasks, but it is not trivial. There are two main formalisms that are used for modeling syntax and subsequently lead to different parsing approaches; one is a constituency formalism, which views grammar as the interaction of groups of words (constituents) behaving as one unit, and the other is a dependency formalism, which describes the syntax of sentence solely in terms of directed binary relationships between words (Jurafsky and Martin, 2009).

1.1 Grammatical Relations

Grammatical relations (GRs) (Carroll et al., 1998) consist of a head and a dependent, and they also specify the grammatical function that the dependent plays with respect to the head. In traditional grammar, these are functions like subject, direct object, and indirect object, etc. but many modern parsers use a much fuller taxonomy of relations. Jurafsky and Martin (2009) describe two broad categories of grammatical relations: clausal relations that describe syntactic roles with respect to a predicate, and modifier relations which categorize the ways that words can modify their heads. An example of the sentence "Kim handed Sandy two large boxes" parsed into its grammatical relations can be seen in Figure 1.

[1]Full codebase as well as raw parser output is available at `github.com/hmcgovern/parser-eval`



Example: Kim handed Sandy two large boxes
Grammatical Relations:

(ncsubj handed Kim)
(dobj handed Sandy)
(obj2 handed boxes)
(amod boxes large)
(nummod boxes two)

Figure 1: Example of grammatical relation output

Different parsers use different taxonomies of GRs and format them differently. Some schema are more fine-grained than other (for example, Briscoe et al., 2006's RASP system uses a much more granular set of GRs than the Universal Dependencies (Nivre et al., 2016)), which is an obstacle to overcome when attempting a standardized comparison between these parsers. The two parsers we evaluate use different output styles of GRs: SpaCy reports GRs with CLEAR-style dependencies while the Stanford parser reports GRs with Stanford-style dependencies. We discuss the issue of accurate comparison between different styles of GRs more in Section 5.

1.2 Overview of Paper

The goal of this paper is to evaluate two off-the-shelf parsers with different underlying parsing algorithms on a curated test set of sentences and subsequently to draw general conclusions about the strengths and weaknesses of each parser. Evaluated in this paper is the SpaCy CNN-based parser (Honnibal et al., 2020) and the Stanford PCFG parser (Klein and Manning, 2003). We will first introduce each parser and its specific preprocessing requirements, then introduce the example sentences (included in full in Appendix A) and discuss likely challenges aspects of these test sentences will pose for the parsers. Then, we will present the concept of unbound dependencies as a quantitative metric by which to evaluate the parsers as described by Rimell et al. (2009) and discuss the process of curating a partial gold-standard set of unbound dependencies. We will describe our experimental design and then, finally, we will report results of these experiments, compare the overall output of each parser in consideration of the complete qualitative and quantitative evaluations, and discuss the strengths and weaknesses of our approach.

2 The Parsers

2.1 SpaCy

SpaCy is a library for natural language processing in Python. The library allows the user to download a number of pre-trained neural models to use off-the-shelf. We choose to evaluate the *en_core_web_md* model, which is a multi-task convolutional neural network (CNN) model trained on English OntoNotes 5 and also incorporates 300-dimensional GloVe vectors, which themselves were trained on Common Crawl. The main distinction between this model and the most basic pretrained model included in the SpaCy library (*en_core_web_sm*) is the preference for representing words with contextualized word vectors rather than as a static vector. The model is trained to predict named entities, part-of-speech tags, and syntactic dependencies given raw text which is first passed through a tokenizer.

Notably, the individual components of the SpaCy pipeline like the tagger or parser are independent and do not share any data between themselves. For example, the named entity recognizer does not use any features identified by the tagger and parser, and so on. Instead, the multi-task CNN directly learns how to perform each task in an end-to-end manner.

2.2 Stanford PCFG

The Stanford Probabilistic Context-Free Grammar (PCFG) Parser was introduced by Klein and Manning (2003) as a fast and accurate unlexicalized parser. The underlying framework of a Context-Free Grammar (CFG) is one of the major formalisms used for modelling constituency and consists of a number of distinctive elements: a set of terminal symbols that are the words of the language, a set of non-terminal symbols which are abstractions over the terminal symbols, a designated start symbol ('S'), and a set of production rules which dictate the allowed groups and orderings of terminals and non-terminals (Jurafsky and Martin, 2009). A probabilistic CFG additionally provides a probability associated with each production rule specifying how likely that rule is to be called. These probabilities are calculated from the training corpus, which for the Stanford PCFG is the Wall Street Journal (WSJ) corpus. To parse such a grammar, the Stanford PCFG uses an implementation of the generalized CKY algorithm, which uses a table of partial parses to efficiently parse ambiguous sentences (Klein and Manning, 2003). A key aspect of this parser is that it is unlexicalized; it does not place much emphasis on the structure of the lexicon, which lessens the burden on pure phrase-structure rules.

The pipeline for the Stanford PCFG is as follows: raw text is first tokenized, then split into sentences, tagged with parts-of-speech tags, lemmatized, mined for Named Entities (NEs), and finally parsed for dependencies.

3 The Test Sentences

We use a subset of the provided example sentences for qualitative evaluation as well as a number of extra test sentences added to the original test set in order to highlight certain behaviors of the parsers in the quantitative evaluation. Why and how these sentences were chosen to be added is addressed more fully in Section 4.2. In this section, we will be flagging constructions and fragments we expect to be handled poorly by the parsers. In Section 7, we will discuss in detail how each parser actually handled these supposed issues, along with any other noticeable errors.

3.1 Rich-Text Formatting

Many of the sentences contained semantic information encoded in rich-text formatting, such as *italics*, subscripts/superscripts, or enclosed in en-dashes (–), whose emphasis is lost when converted to plain text.

For example, Sentence 17 in the provided test sentences uses italics to place emphasis on the elements of a list whose elements happen to be modal verbs:

> "…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*."

Similarly, Sentence 18 contains mathematical notation which uses subscripts:

> "a sequence of observed words $O = (o_1, o_2, ...o_T)$ and returns the most probable state/tag sequence $Q = (q_1, q_2, q_T)$ together with its probability."

3.2 Relative Clauses

When a subject or object is extracted from the main level of the sentence into a relative or subordinate clause, there is a high chance that a parser will not be able to recover the grammatical dependency, especially if the dependent and the head are very distant from each other in the sentence.

Sentence 9 incorporates nested relative clauses, shown below in square brackets, introduced by the relative pronouns *who* and *that*:

> "The veterans [who I thought [that we would meet at the reunion]] were dead."

In this sentence, "veterans" is the extracted object of the verb "meet", as is made more clear if one were to write a related fragment, *the veterans that we met at the reunion* or *I thought that we would meet the veterans.* In this case, the parser must extract the clausal argument across a double boundary.

3.3 Coordination

We use this term to refer to two kinds of constructions: (1) those with coordinated subjects and (2) those with coordinated action verbs. For example, in Sentence 4, the subjects "Kim" and "Sandy" are coordinated, meaning they are both arguments of the same verb:

> "**Kim** and **Sandy** both broke up with their partners" (emphasis added)

"Kim and Sandy" could be replaced by "they" without changing the meaning of the sentence. The two subjects are conjuncts.

In Sentence 23, one subject performs multiple actions in parallel:

> "...But far fewer people fully understand how the Media Lab **operates**, **fits** into MIT, and **encourages** such a creative environment" (emphasis added)

A correct parse would indicate that 'lab' is the non-clausal subject of all three verbs.

4 Unbound Dependencies

4.1 Constructions

Rimell et al. (2009) describe an evaluation method for grammatical parsers using seven types of "unbound dependencies," which they define as constructions in the text which "[contain] a word or phrase which appears to have been moved, while being interpreted in the position of the resulting 'gap'." The fact that there may be an arbitrary distance between the head and the dependent is what makes these kinds of dependencies "unbounded." This method of evaluation was designed to compensate for the weaknesses of the popular but deprecated PARSEVAL script (Black et al., 1991); namely, by reporting a single score, PARSEVAL and similar scripts (such as CoNLLEval, MALTEVAL, etc.) smooth out differences in recovering dependency relations, which are not equally difficult to retrieve. For example, in Sentence 4, "Kim and Sandy both broke up with their partners," the subject coordination of "Kim and Sandy" is a significantly harder relation for a parser to recover than the objection relationship of "broke" and "partners" but is just as semantically important to the sentence.

As many such examples of unbounded dependencies are difficult to recover with only shallow parsing techniques, we instead choose to focus only on the recovery of these "hard" dependencies. This arguably provides a stronger test of grammar representation than a mere survey of accuracy over all reported dependencies. We like this method for the straightforwardness of its approach and, because all gold standards are created by a single annotator, it reduces the amount of time required to create a gold label set to only annotate certain important dependencies.

Here we describe each of the seven types of unbound dependencies which we aim to identify in our test sentences for evaluation:

1. **Object extraction from a relative clause** a relative pronoun introducing a clause from which an object argument has been extracted: e.g. *...the application that I submitted.*

2. **Object extraction from a reduced relative clause** almost identical to the above but the relative pronoun is not-present: e.g. *...the paper I wrote.*

Construction	**Frequency**		**Distance**	
	Small Set	Full Set	Avg. Dist	Max Dist
Obj rel clause	0.50	0.20	3.86	8
Obj reduced rel	0.10	0.17	3.75	6
Sbj rel clause	0.30	0.17	5.67	13
Free rel	0.0	0.13	3.00	5
Obj wh-Q	0.0	0.17	7.25	7
RNR	0.0	0.13	2.50	5
Sbj embedded	0.10	0.13	4.50	6

Table 1: Frequency of constructions in the smaller original set and the full set as a percentage of sentences in the corpus as well as average and maximum distance between head and dependent relation in the full set

3. **Subject extraction from a relative clause** a relative pronoun introducing a clause from which a subject argument has been extracted: e.g. *The idea that I had was profound.*

4. **Free relatives** relative pronouns without antecedents: e.g. *I wonder what she said.*

5. **Object *wh*-questions** a *wh*-word is the semantic object of the verb: e.g. *What time is your class?*

6. **Right node raising (RNR)** coordinating phrases from which a shared element apparently moves to the right: e.g. *. . . questions about or information pertaining to the procedure . . .*

7. **Subject extraction from an embedded clause** a semantic subject which is apparently extracted across two clause boundaries: e.g. *the necklace which the maid said she hadn't seen . . .*

4.2 Creating a Gold-Standard

For a quantitative metric to be meaningful, we need to have a gold standard against which to compare. To this end, we identify a collection of unbound dependencies of the types described in Section 4.1 and evaluate our parsers on their ability to recover those relations. If we look at just the collection of sentences taken from the provided sentences, as provided in Table 1, we see that some constructions are much more frequent than others. To remedy this, we add a number of example sentences such that the occurrance frequency of each construction in the full set is roughly the same. The additional sentences added were taken from Rimell et al. (2009)'s examples of the dependencies, along with some we created to round out the least frequent categories. In total, there are 30 distinct sentences in the full set and 35 instances of unbound dependencies, with the frequency metrics as report in Table 1. The full set of example sentences used for the quantitative evaluation and gold standard dependency relations can be found in Appendix B.

4.3 Differences in GR Schema

As noted in Section 1.1, the two parsers we use output GRs according to different schema. This makes a direct comparison to a gold standard difficult to automate, but considering the small size of the dataset, it was feasible to manually compare the output for each label. We considered a correctly returned dependency one that recovered the correct head, correct dependent, and an acceptable match for the GR itself. For example, if the gold standard GR is 'dobj' but a parser returns merely 'obj' we count that as correct. Similarly, if the gold standard is 'prep' and the parser returns a more fine-grained 'prep:in', we also count that as correct.

Both the Stanford and SpaCy parser use a tag for a relative clausal modifier, 'acl:relcl' and 'relcl', respectively. While the taxonomy of the unbound dependencies make a more fine-grained distinction between the subject or object extraction in the relative clause, neither of our parsers do. In these cases, we consider it correct if the parser has identified the correct head and dependent with relative clausal modifier tag. However, if the parser identified the correct head and dependency but listed the relation as 'dep', an unspecified dependency, that was not counted as correct.

5 Experimental Design

After the parsers were installed and compiled, they were essentially run out-of-the-box on a CSV file containing all the test sentences formatted in plain text. The Stanford PCFG parser was run through a command line tool and its output piped to a text file. SpaCy has a useful API for Python that was used to load a pre-trained statistical parser which then was run over each sentence individually. We then write the output to a file in a format approximating the Stanford style for easier visual comparison.

The Stanford PCFG package included separate models for uncased and cased English text, so we pre-processed the input text to be all lowercase and used the uncased model. For consistency's sake we then used lowercase text as input for SpaCy as well. No extensive pre-processing was done other than this.

6 Results

The accuracy scores for the two parsers can be seen in Table 2. The higher score in each column is marked in bold. For objects extracted from relative clauses, SpaCy outperforms the Stanford parser with an accuracy of 83.3% compared to 50%. For objects extracted from a reduced relative clause, both parsers achieve a 60% accuracy. For subjects extracted from a relative clause, SpaCy outperforms the Stanford parser with 60% compared to 40%. For free relatives, the Stanford

Parser	**Obj RC**	**Obj Red**	**Sbj RC**	**FR**	**wh-Q**	**RNR**	**Sbj Embed**	**Total**
SpaCy	**83.3**	**60.0**	**60.0**	50.0	**40.0**	25.0	25.0	51.5
Stanford	50.0	60.0	40.0	**100**	25.0	**80.0**	**50.0**	**57.8**

Table 2: Parser accuracy on unbound dependencies along with macro accuracy

parser outperforms SpaCy with 100% compared to 50%. For object *wh*-questions, SpaCy outperforms the Stanford parser with 40% compared to 25%. For right node raising, the Stanford parser outperforms SpaCy with 80% compared to 25%. For subjects extracted from embedding clauses, the Stanford parser outperforms SpaCy with 50% compared to 25%. Overall, the Stanford parser had a slightly higher total accuracy of 57.8% compared to SpaCy's of 51.5%.

Overall, the quantitative results do not point to a obvious winner. The Stanford parser dominates in constructions that translate well to tree-like structures such as free relatives (which would be represented with an SBAR in a PST) or right node raising (which is inherently tree-like), but SpaCy dominates in extracting clausal arguments from relative clauses. When we consider the head-dependent distance in Table 1, there is no clear correlation between the average distance of the construction and the accuracy of a certain parser.

7 Discussion

In this section, we analyze the errors using the categories we described in Section 3 and attempt to explain why they occurred according to our understanding of the underlying grammars on which the parsers are based. Some errors could be as more than one type of error but we aim to place it in the most relevant category. Additionally, we will compare the accuracy metrics obtained by our unbound dependency evaluation in light of the respective speeds of the parsers. Finally, we will discuss the merits and limitations of our approach.

It is important to highlight that the pipelines of the two taggers are substantially different. The SpaCy parser is end-to-end whereas the Stanford PCFG parser uses features such as POS and NER which may be responsible for propagating classification errors.

7.1 Text Formatting Errors

1. In Sentence 10, the fragment '– storms, flooding, and hurricanes –' functions as an appositional modifier to the noun phrase 'Natural disasters'. The Stanford parser is able to correctly identify this appositive as a unit with the correct relationship to the main subject, but fails to recover the subject relationship of 'disasters' to the verbs 'occur' and 'cause', even though it does recognize that 'occur' and 'cause' are conjuncts of each other. If we replace the en-dashes with parentheses, the Stanford parser is able to correctly recognize it as a full sentence in which 'disasters' is the subject of 'occur' and 'cause'. The Stanford parser is able to parse the fragment

as a unit because of its underlying phrase-structure grammer. SpaCy, on the other hand, does not make use of tree-like syntactical structures and therefore the parsing is quite flat; it identifies 'hurricanes' as the subject of the verb 'occur' because it is the closest noun to the verb. Additionally, SpaCy correctly identifies 'storms' as an appositional modifier of 'disasters', but then thinks that 'flooding' is the conjunct of 'storms' instead of another appositional modifier of 'disaster' because it does not take the constituency of the dashed fragment into account.

2. In Sentence 17, modal verbs *do, did, doing, be* and *have* appear as nouns in italics. As this rich-text formatting is lost upon input to the parsers, both incorrectly mark these entities as verbs. Wrapping them in quotations marks does not ameliorate the issue for either parser. It's not clear that one parser has an advantage over the other for this particular case.

3. In Sentence 18, the mathematical notation produces errors for both parsers. The Stanford parser tags the algebraic terms '$o_1, o_2, ...o_T$' as cardinal numbers within a parenthetical phrase which does not include '$O =$', whose relationship it deems a case marker. Additionally, the parser shows unclassified dependency ('dep') between the elements of the o sequence, but a numeric modifier ('nummod') between the elements of the q sequence. Interestingly, where the Stanford parser labels all algebraic elements as cardinal numbers (we think they are better classed as generic nouns), SpaCy labels 'o1' and 'q1' as numbers, 'o2' and 'q2' as proper nouns, and 'oT' and 'qT' as generic nouns, ostensibly projecting the fragment onto some pattern it had learned in training to represent elements in a list. Similarly to the appositional phrase in Sentence 10, the mathematical notation is sectioned out better by the Stanford parser because its grammar is tree-based.

7.2 Coordination Errors

1. In Sentence 6, the fragment 'as well as' is tagged by SpaCy as a subordinating conjunction when it should be part of a coordinating phrase because it could simply be replaced with 'and'. The Stanford parser correctly identifies this as a coordinating phrase, demonstrating the advantage of converting dependency relations from PSTs. Both parsers are able to recover the conjunct relationship of 'horse' and 'rabbits' as well as the relationship of 'rabbits' and 'wanted' in the relative clause, but neither identifies 'horse' as the other object of the verb 'wanted'. SpaCy tends to associate the nearest noun with the verb, even across clause boundaries, which may explain why it misses this relationship. The Stanford parser's PST structure also is not helpful for recovering this coordination because the relative clause containing the fragment 'which we wanted to eat' branches exclusively from the noun phrase 'the rabbits'. This is a weakness of the Stanford parser with respect to coordinated subject and relative clauses.

2. In Sentence 17, the Stanford parser recovers the three-fold coordination between the list elements ‘do’, ‘be’, and ‘have’ even though there is a 15 word separation between the first two items in the list because they are separated by the parenthetical phrase ‘(e.g. C5 tag VDD for *did* and VDG tag for *doing*)’. SpaCy only recovers the conjunct relationship between ‘be’ and ‘have.’ Routinely when parsing lists, SpaCy seems to only recognize conjuncts when it fits the pattern ‘X and Y’ exactly. Even if we modify the input to remove the parenthetical phrase, SpaCy still only recognizes the last two items of a list as conjuncts.

3. In Sentence 18, the Stanford parser correctly recognizes ‘version’ as the subject of both ‘takes’ and ‘returns’ as well as the conjunct relationship between the two, whereas SpaCy tries to coordinate ‘O_T’ with ‘returns‘ because they are right next to each other. As SpaCy has no built-in concept of constituency, it routinely fails when dealing with appositional or parenthetical phrases. Rather, because it is based on a CNN, which learns spatial dependencies in the data with convolutional filters, it provides a much flatter parse. Its contextualized word embeddings are not likely to be of much help with rare words such at ‘O_T’.

4. In Sentence 20, in the fragment ‘from great personal success, or just an all-night drive, we. . . ’ a parser should be able to recover the coordination between ‘success’ and ‘drive’, recognizing that ‘from’ is a preposition relating to both of them (‘from success’ and ‘from [a] drive’). Neither parser associated ‘drive’ with its preposition, although SpaCy correctly identifies ‘success’ as the object of the preposition ‘from’. This is in line with Spacy’s tendency to only identify the nearest object of a coordinated phrase. However, Stanford identifies the relationship of ‘from’ and ‘success’ as a case marking when it is not. Stanford correctly recovers the oblique nominal relationship between ‘survivors’ and ‘success’ as well as ‘survivors’ and ‘drive’.

5. In Sentence 23, the three-fold coordination of ‘...the media lab operates, fits . . . , and encourages’. The Stanford parser is able to recover the conjunct relationship between ‘operates’ and ‘fits’ as well as ‘operates’ and ‘encourages’, spanning the three verbs; whereas, SpaCy is only able to recover the conjunction between ‘fits’ and ‘encourages’, which are tied together explicitly with the coordinating conjunction ‘and’. Consequently, SpaCy cannot correctly identify the subject of the verbs, which is ‘lab’ for all three, instead identifying ‘media’ as the subject of ‘operates’. It is once again evident the power that the Stanford parser’s underlying PST structures holds to recover coordinated constituents while SpaCy’s CNN structure only handles linear dependencies.

7.3 Relative Clause Errors

1. Sentence 16 deals with coordination within a relative clause: ‘heuristics that help’ and also ‘that provide.’ Both parsers correctly identify ‘provide’ as the conjunct of ‘help’, but only SpaCy detects the subject extraction to the relative clause. SpaCy’s CNN base makes its parse less dependent on clause boundary markers than the Stanford parser is. However,if we simplify the sentence to, ‘tagging manuals give various heuristics that can help human coders make these decisions and also provide useful features for automatic taggers.’ the Stanford parser can not only detect the presence of the relative clause, but also detect both the relationships ‘heuristics help’ and ‘heuristics provide’. In this case, the underlying phrase-structure system of the Stanford parser actually hurts its performance as the additional ‘that’ caused an extra branch containing that relative clause to split off of the PST. Because the Stanford parser uses a context-free grammar, once the PST has been developed, there is no communication between different branches. SpaCy is not bothered by this, and can either way detect the first relative clause just by virtue of their proximity in the sentence, but the modified sentences does not help it recover the coordination, and actually instead tried to list ‘make’ as the conjunct of ‘provide’ because they are closer together. This shows that SpaCy places priority on recognizing patterns in the text such as the pattern ‘X and Y’ as conjuncts.

2. In Sentence 18, SpaCy recovers the fact that ‘version’ is the object of the phrase ‘that we present’ in a relative clause. The Stanford parser treats the entire phrase ‘of the Viterbi . . . present’ as a prepositional phrase modifying ‘version’ and so does not detect the relative clause. This is an example where a slightly different phrase structure tree obfuscates the dependency parsing where SpaCy’s linear process succeeds.

3. Sentence 20 contains a reduced relative, i.e. a relative clause that is missing the relative pronoun introducing it: ‘a world [which] no one has ever seen.’ Neither parser is able to recover this as a reduced relative clause, but if we modify the sentence to make the ‘which’ explicit, the Stanford parser recovers it while SpaCy does not.

7.4 Parsing Speed

For our full test set comprising 30 number of sentences, SpaCy completed parsing in 0.275 seconds, (2,534 words/sec), but loading the model prior to parsing takes an additional 3.69 seconds. To train your own SpaCy model would require extra time as well.

The Stanford parser only takes 0.7 seconds to load, but 18.4 seconds to parse the same sentences (26.43 words/sec). The Stanford model we test is the fastest version of the English PCFG parser; Stanford also offers a factored product model which uses the A* algorithm to combine the inferences of a separate

PCFG phrase structure and lexical dependency experts. However, that parser is slower with only marginally better accuracy (Klein and Manning, 2002).

Even accounting for SpaCy's loading time, which only has to be done once in the namespace, it is lightning fast due to its efficient Cython implementation (Honnibal et al., 2020). A slow load-time parser but fast run-time parser like SpaCy is ideal for scalable applications or very large datasets. The Stanford parser, which is a fast load-time but slow run-time parser, is better for research contexts or smaller datasets.

7.5 Limitations of Our Approach

One limitation of our approach to quantitative evaluation is the very small size of the dataset used, with our full evaluation set only comprising 30 sentences. In most of the parser evaluation literature, a minimum of a few hundred sentences would be considered necessary to draw reliable conclusions about a parser's accuracy (Preiss, 2003; Rimell et al., 2009; Briscoe and Carroll, 2006). If we had used an approach similar to PARSEVAL where we fully annotated every sentence and created some type-wise precision, recall, and F1 score, we would have had more examples as each sentence contains many dependencies but only 1 or 2 unbound dependencies. However, with more granular annotation comes the possibility of greater annotator error, and as only one annotator was responsible for producing the gold-label set, our approach of only comparing 'hard' dependencies was justifiable. The major strength of our approach is that it places a high priority on how well parsers handle difficult situations and places less emphasis on easier cases. Therefore our approach more directly provides a strong test of the parser's understanding of the grammar of a language, hampered as it is by the small evaluation set.

Finally, one limitation that emerged when doing the quantitative comparison was that the Stanford parser was able to recover the correct heads and dependents in many situations where SpaCy was not, but it tagged them as generic 'dep' and so were not counted as correct. So, the quantitative results do not fully reflect the difference in accuracy of the two parsers the way that an unlabeled and labeled accuracy score (LAS and UAS) used in different evaluation methods might.

8 Conclusion

We have examined the performance of two popular dependency parsers on a set of test sentences. We compared SpaCy, a neural network-based statistical parser, and the Stanford parser, an unlexicalized phrase-structure based parser. We both qualitatively examined output and quantitatively measured the parsers' ability to recover a number of potentially long-range dependencies in the test sentences.

Overall, SpaCy is the faster but less qualitatively accurate parser. Its efficient implementation in Cython and lightning-fast processing speed is what renders it a useful tool for industrial use, which is how SpaCy markets itself. In the qualitative assessment, SpaCy seemed to under-perform compared to the Stanford parser; however, in a quantitative comparison, the two parsers were not very different in overall accuracy. While SpaCy does have the ostensible advantage that errors in other stages of processing like NER or POS-tagging cannot be propagated to the dependency parser, it lacks the underlying phrase-structure tree system that enable the Stanford parser to group parentheticals and appostives into constituents. CNNs leverage spatial dependencies in the data, which in SpaCy's case often looks like recognizing patterns like 'X and Y', whereas the Stanford parser actually captures syntactic relationships on a larger scale because of the constituency formalism on which it is based. The Stanford parser's major drawback is that once a PST is derived in a context-free grammar, there is no communication between nodes. As a result, if a relative or subordinate clause is parsed into a tree incorrectly, it will make many subsequent dependency parsing errors. SpaCy, on the other hand, can take advantage of spatial relationships between the words because it uses a CNN as an underlying structure and has some built-in measure of word meaning from its contextualized word embeddings. In conclusion, for a wider swatch of use cases, the Stanford PCFG parser is a superior parser, with only processing speed and occasional clausal argument extraction as points in favor of SpaCy.

References

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991.*

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. (July):41–48.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 77–80, Sydney, Australia. Association for Computational Linguistics.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings, First International Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain. European Language Resources Association.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Dan Jurafsky and James H. Martin. 2009. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J.

Dan Klein and Christopher D. Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, page 3–10, Cambridge, MA, USA. MIT Press.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. pages 423–430.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 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, Portorož, Slovenia. European Language Resources Association (ELRA).

Judita Preiss. 2003. Using grammatical relations to compare parsers. page 291.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. *EMNLP 2009 - Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: A Meeting of SIGDAT, a Special Interest Group of ACL, Held in Conjunction with ACL-IJCNLP 2009*, (August):813–821.

A Test Sentences

5. The horse which Kim sometimes rides is more bad tempered than mine.

6. The horse as well as the rabbits which we wanted to eat have escaped.

7. It was my aunt's car which we sold at auction last year in February.

8. The only rabbit that I ever liked was eaten by my parents one summer.

9. The veterans who I thought that we would meet at the reunion were dead.

10. Natural disasters – storms, flooding, hurricanes – occur infrequently but cause devastation that strains resources to breaking point.

16. Making these decisions requires sophisticated knowledge of syntax; tagging manuals (Santorini, 1990) give various heuristics that can help human coders make these decisions and that can also provide useful features for automatic taggers.

17. 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.
18. 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.

20. Coming home from very lonely places, all of us go a little mad: whether from great personal success, or just an all-night drive, we are the sole survivors of a world no one else has ever seen.

23. But far fewer people fully understand how the Media Lab operates, fits into MIT, and encourages such a creative environment; about half of the anniversary celebration's program focused on simply defining what the Media Lab is.

B Additional Sentences For Quantitative evaluation

The unbound dependency evaluation included all of the sentence in Appendix A as well as the following.

B.1 Object extraction from a reduced relative clause

- By Monday, they hope to have a sheaf of documents both sides can trust.
- Each must match Wisman's "pie" with the fragment he carries with him.
- The slightly simplified version of the Viterbi algorithm 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.
- The veterans who I thought we would meet at the reunion were dead.

B.2 Subject extraction from a relative clause

1. It consists of a series of pipes and a pressure-measuring chamber which record the rise and fall of the water surface.

B.3 Free relatives

1. He tried to ignore what his own common sense told him, but it wasn't possible; her motives were too blatant.
2. Nobody knows what happened because no one else was in the room when it happened.
3. She couldn't ignore what her heart was telling her.
4. He told her he needed to find himself, whatever that means.

B.4 Object *wh*-questions

1. What city does the Tour de France end in?
2. What did you submit to your thesis advisor?
3. What did you call to tell me?
4. Where should I park to avoid getting a parking ticket?
5. What did you eat during your vacation in Rome?

B.5 Right node raising (RNR)

1. For the third year in a row, consumers voted Bill Cosby first and James Garner second in persuasiveness as spokesmen in TV commercials, according to Video Storyboard Tests, New York.

B.6 Subject extraction from an embedded clause

1. In assigning to God the responsibility which he learned could not rest with his doctors, Eisenhower gave evidence of that weakening of the moral intuition which was to characterize his administration in the years to follow.
2. Honolulu, which is the center of the warning system, was under threat of attack
3. The stranger who had held out his cigarette lighter to me retreated back into the shadows
4. The veterans who I thought would be at the reunion were dead.