The Effects of Disfluency Detection in Parsing Spoken Language

Fredrik Jørgensen
Department of Linguistics and Scandinavian Studies
University of Oslo
fredrik.jorgensen@iln.uio.no

Abstract

Spoken language contains disfluencies that, because of their irregular nature, may lead to reduced performance of data-driven parsers. This paper describes an experiment that quantifies the effects of disfluency detection and disfluency removal on data-driven parsing of spoken language data. The experiment consists of creating two reduced versions from a spoken language treebank, the Switchboard Corpus, mimicking a speech-recognizer output with and without disfluency detection and deletion. Two data-driven parsers are applied on the new data, and the parsers' output is evaluated and compared.

1 Introduction

Spoken language data differs from written language data in several respects. Spoken language is less "well-behaved" or well-formed than written language in the sense that in spoken language, all editing is performed real-time. There is no possibility of deleting what has been said, and utterances often contain repetitions, corrections and interruptions, or are left unfinished. These phenomena are often referred to as disfluencies, and their status in syntax is unclear. However, in syntactically annotated spoken language corpora (i.e., treebanks), disfluencies are often retained as unconventional syntactic patterns, in contrast with the rest of the annotation, which is solely syntactically motivated.

This experiment seeks to quantify data-driven parser performance on spoken language data with and without disfluencies. The hypothesis for the experiment is that because data-driven parsers generalize over regularities in language, and because language containing disfluencies is expected to be less regular than more well-formed or grammatical language, the detection and removal of disfluencies will improve parser performance.

The result of this study indicates to what degree of magnitude disfluencies affect parser performance, and what kind of performance gain can be expected when removing disfluencies prior to parsing. The study may also be used as a partial basis for deciding when disfluency detection is more effective, either prior to or during parsing.

For this experiment, I have chosen data-driven parsers as opposed to manually written, rule-based parser, because most rule-based parsers have no way of handling disfluencies. It is very difficult to formulate general rules about the syntactic structure of disfluencies, and it is not clear whether these rules actually are syntactic rules or merely generalizations over how we communicate or perform online editing of utterances. This in turn means that disfluency detection may be crucial for a rule-based parser applied on spoken language.

2 The Experiment

2.1 Experiment Outline

The main idea of this experiment is to take a treebank, transform it into versions with and without disfluencies, and apply data-driven parsers on the new versions.

The treebank chosen in this experiment is the

Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek and Mare Koit (Eds.)
NODALIDA 2007 Conference Proceedings, pp. 240–244

Data Set	*Sentences*	*Words*	*Files*
Training	46104	604967	sw2005 - sw3993
Test	6998	81787	sw4004 - sw4936
Total	53102	686754	sw2005 - sw4936

Table 1: Description of Training and Test Data

Switchboard Corpus. This treebank has a consistent annotation of disfluencies, which makes the disfluencies easy to identify and remove, without taking any particular view on the status of disfluencies in relation to syntactic theory. We transform the treebank into three different versions. One *Original* version for comparison, one *No Markup* version, where all punctuation and disfluency markup is removed, and one *No Disfluency* version, where all disfluencies are removed. This is described in more detail in Section 2.3.

After the transformation, two data-driven parsers (one constituency-based and one dependency-based) have been trained and tested on the new versions of the treebank. Finally, the parser performance on the different versions are evaluated and compared.

2.2 Prerequisites

2.2.1 Data: The Switchboard Corpus

The Switchboard Corpus, a part of the Penn Treebank (Marcus et al., 1993), consists of recorded telephone dialogs. The corpus is annotated syntactically, with specific node labels and Part of Speech (PoS) tags for speech related phenomena. These include:

- Fillers and discourse markers (UH)
- Unfinished nodes (XP-UNF)
- Edited sections (EDITED) Following the terminology of Shriberg (1994), an EDITED node consists of the *reparandum*, and is immediately followed by a *repairs*, the string replacing the reparandum.

An example of a tree from the Switchboard Corpus is given in Figure 1.

The data used for training and testing is from the parsed version of the Switchboard Corpus, as described in Table 1.

2.2.2 Parsers

Two parsers were trained and tested on the three versions of the treebank: Dan Bikel's Parser (constituency-based) and the MaltParser (dependency-based). The choice of *particular* parsers is not central to this experiment, as we are comparing different data, and not different parsers. But by including one constituency-based and one dependency-based parser, we are able to see if the same tendencies apply to both *types* of parsers.

Dan Bikel's parser[1] is based on the parsing algorithm of Michael Collins' head-driven lexicalized statistical parser (Collins, 1999).

The MaltParser[2] reduces the parsing process to a classification task, using a machine learning method of choice (memory-based learning or support vector machines) for classification. For details, see Nivre and Hall (2005). Before applying the MaltParser on the data, it had to be converted from constituency annotation (labeled bracketing) to dependency relations (labeled relations between words). This was done using the Penn2Malt[3] converter.

2.3 Treebank Transformation

As mentioned above, the Switchboard Corpus was transformed into three different versions:

1. An *Original* version. Nothing is removed.

2. A *No Markup* version, where all non-word markup is removed. This includes punctuation, traces and disfluency markup (e.g. interruption points, start and end tags of repair sequences etc.). This transformation is intended to mimic the output from a speech recognizer.

   Removed: -DFL- -NONE- : , . ”

3. A *No Disfluencies* version, where all repairs, unfinished nodes and parenthenticals are removed. In addition, all sentences ending in an unfinished node are considered unfinished sentences, and are also removed:

   Removed: *xp*-UNF EDITED PRN

[1] http://www.cis.upenn.edu/~dbikel/software.html
[2] http://www.vxu.se/msi/~nivre/research/MaltParser.html
[3] http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

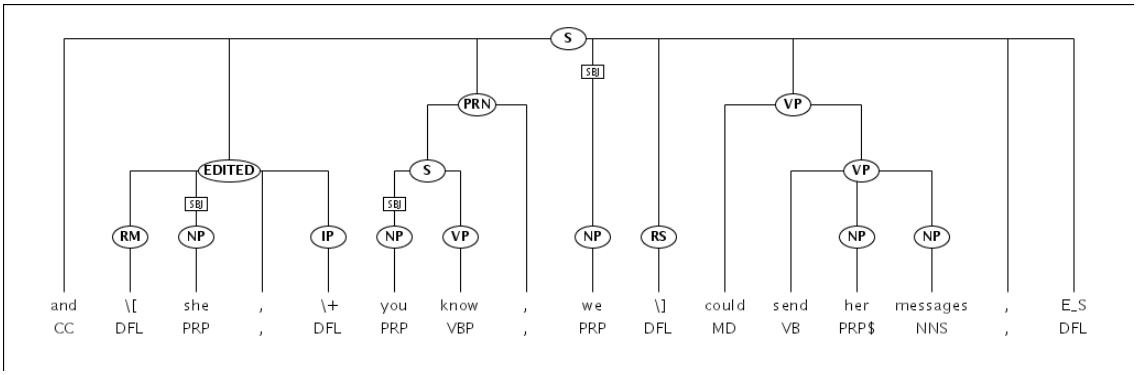

Figure 1: Original sentence from the Switchboard Corpus

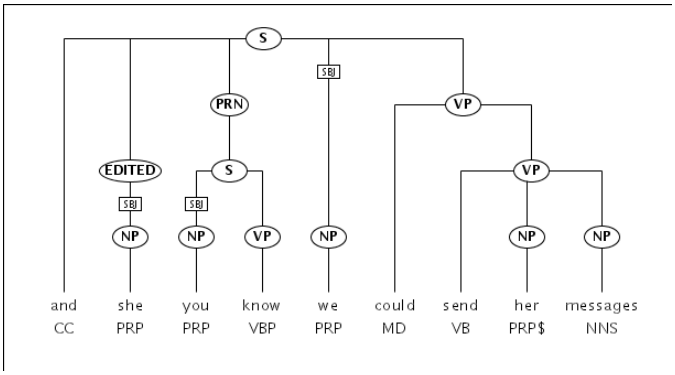

Figure 2: Sentence from the Switchboard Corpus, *No Markup* version

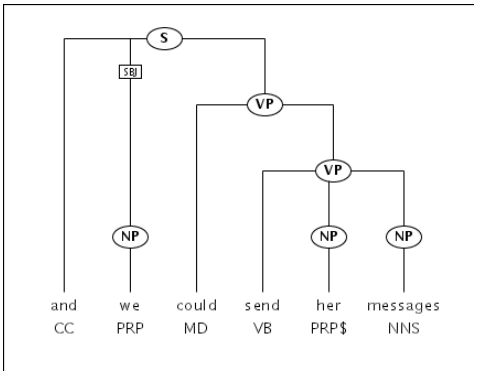

Figure 3: Sentence from the Switchboard Corpus, *No Disfluencies* version

Corpus Version	*Avg. Words per Sentences*
Original	11.69
No Markup	9.93
No Disfluencies	9.12

Table 2: Avg. sentence length after transformation

In addition, all sentences consisting of 2 or less words were removed from the corpus. The average sentence length after filtering is shown in Table 2.

The transformation was performed by a perl script, available at `folk.uio.no/fredrijo/software`. The *Original* sentence in Figure 1 can be seen as *No Markup* in Figure 2 and *No Disfluencies* in Figure 3.

2.4 Running the Experiment

The experiment consists of the following steps:

1. **Transformation:** Transform the treebank into three different versions.
2. **Filtering:** Filter out sentences with two words or less in the *No Disfluencies* version from all versions of the treebank.
3. **Conversion**: Convert the treebank to Malt Tab format, using Penn2Malt (for the MaltParser only).
4. **Training:** Train the parser on each of the three versions.
5. **Parsing:** Parse the test collection, using the three different "versions" of the parser.
6. **Evaluation:** Evaluate the parser output against the manually annotated gold standard test set. For the constituency parser, the evaluation metrics are precision, recall and F score for labeled and unlabeled bracketing. The scores are generated using `evalb`[4]. For the dependency

[4]http://nlp.cs.nyu.edu/evalb/

parser, the evaluation metrics are labeled and unlabeled attachment (of dependency relations) and label accuracy (for the edge labels), using the CONLL evaluation script `eval.pl`[5].

3 Results

The test set of 6998 sentences was divided into 10 partitions or samples. The results for Dan Bikel's Parser and the MaltParser are shown in Table 3 and Table 4, respectively. The results are given as sample means with standard error.

As we see from the results for Dan Bikel's Parser, there are statistically significant differences in results between the three versions of the corpus. A comparison of the *No Markup* and the *No Disfluencies* versions is shown below, with p values:

Lbl. Precision	+1.71	($p < 0.0001$)
Unlbl. Precision	+1.63	($p < 0.0001$)
Lbl. Recall	+1.68	($p = 0.0003$)
Unlbl. Recall	+1.60	($p = 0.0001$)
Lbl. F Score	+1.70	($p < 0.0001$)
Unlbl. F Score	+1.62	($p < 0.0001$)

The results for the MaltParser are somewhat different, as the results for the *Original* version are better than for the *No Markup* version. But if we compare the *No Markup* and *No Disfluencies* versions, we see that the improvement here is also significant, as shown below:

Labeled attachment	+2.16	($p < 0.0001$)
Unlabeled attachment	+1.71	($p < 0.0001$)
Label accuracy	+1.90	($p < 0.0001$)

Note also the MaltParser's decrease in performance from the *Original* version to the *No Markup*, shown in Table 4. I have no explanation why the MaltParser shows a decrease while Dan Bikel's Parser shows an increase here.

4 Discussion

Before investigating the results, it is worthwhile considering the nature of the experiment for a moment. We are not comparing different systems on the same data set, but rather different data sets. Thus, we are in a sense comparing apples and pears. There are two points to be made here. First, one could imagine trying to evaluate the parsers only on the shared part of the sentence in the *No Markup* and *No Disfluencies* versions, i.e. evaluating the results only on the parts of the sentence that do not contain disfluencies. But as we find crossing brackets into the disfluency sections, it is not clear how this could be done practically. Second, it actually does make sense to compare apples and pears in this experiment. The results also give an indication of how well parsers identify disfluencies, and one way of using the results is to argue for or against passing the sentences to the parser as they are as opposed to detecting and removing disfluencies prior to parsing.

We see, in accordance with the hypothesis, that parsing performance increases significantly (in terms of sample means with standard error intervals) when disfluencies are detected prior to the parsing. It has not been tested here if the improvement is due to the data being more grammatical, and consequentially more regular and predictable, in the *No Disfluencies* version, but this seems a plausible explanation.

One factor that may influence the results, by increasing parser performance, is the fact that the average sentence length is reduced by 2.57 words from the *Original* to the *No Disfluencies* version. I have not tested the significance of sentence length in this study, but this should be investigated.

This experiment mimics change in parsing performance on the output of a "perfect" disfluency detection system. A natural extension to this experiment is to compare the results from a pipe-line system consisting of a disfluency detection system and a parser, to a system where the parser itself is responsible for detecting the disfluencies. Charniak and Johnson (2001) reports one such experiment, concluding that disfluency detection prior to the parsing does not improve parsing significantly. This of course depends on the quality of the disfluency detection as well as how the disfluency detection and the parser integrate. This study, however, states that disfluency detection, when combined with *disfluency removal* prior to the parsing, does have a significant impact on parser performance.

[5] http://nextens.uvt.nl/conll/software.html

Corpus version	Precision		Recall		F Score	
	Lbl.	Unlbl.	Lbl.	Unlbl.	Lbl.	Unlbl.
Original	82.70 ± 0.26	84.94 ± 0.23	82.53 ± 0.29	84.76 ± 0.26	82.62 ± 0.27	84.85 ± 0.25
No Markup	88.45 ± 0.23	89.89 ± 0.21	87.91 ± 0.26	89.34 ± 0.24	88.18 ± 0.24	89.61 ± 0.22
No Disfluencies	90.17 ± 0.22	91.52 ± 0.19	89.59 ± 0.27	90.94 ± 0.24	89.88 ± 0.24	91.23 ± 0.21

Table 3: Results for Dan Bikel's Parser, Sample Means with Standard Errors

Corpus version	Labeled attachment	Unlabeled attachment	Label accuracy
Original	85.99 ± 0.35	88.79 ± 0.32	87.71 ± 0.32
No Markup	85.06 ± 0.27	88.35 ± 0.22	86.65 ± 0.21
No Disfluencies	87.20 ± 0.18	90.06 ± 0.15	88.55 ± 0.16

Table 4: Results for Malt Parser, Sample Means with Standard Errors

5 Conclusion

In this paper, I have quantified the effects disfluency detection and disfluency removal have on data-driven parsing of spoken language data, using two different parsers, Dan Bikel's Parser and the MaltParser, applied on various versions of the Switchboard Corpus. The experiment shows that parsing performance is increased when disfluencies are removed prior to parsing.

References

Eugene Charniak and Mark Johnson. Edit detection and parsing for transcribed speech, 2001.

Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

J. Nivre and J. Hall. Maltparser: A language-independent system for data-driven dependency parsing. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories*, Barcelona, 9-10 December 2005.

E.E. Shriberg. *Preliminaries to a Theory of Speech Disfluencies*. PhD thesis, University of California at Berkeley, 1994.