Improving the Performance of the Link Parser

Y. Viswanatha Naidu, Anil Kumar Singh, Dipti Misra Sharma, Akshar Bharati
Language Technologies Research Centre
IIIT, Hyderabad, India
Email: {vnaidu, anil}@research.iiit.ac.in
{dipti}@iiit.ac.in

***Abstract*—The paper describes an approach to extend the coverage of a Link Grammar based parser on the constructions that are not being handled currently by the grammar. There are about thirty types of constructions which we have identified till now. In order to make Link Grammar handle these constructions, we introduce a preprocessor and a postprocessor. The idea is to handle such constructions via some analysis and transformations in a preprocessing phase before the sentence is given to the Link Parser and then by adding the missing links in the postprocessing phase. The main part of the paper discusses the constructions not handled by the parser and introduces rule based preprocessor and postprocessor. This simple and flexible approach is able to increase the coverage of the parser significantly and allows even a relatively naive user to improve the performance of the parser without disturbing the core grammar.**

I. Introduction

Parsers play a major role in Machine Translation (MT) and, more generally, in a number of other Natural Language Processing (NLP) tasks like question answering systems, anaphora resolution etc. Accordingly, they should be sufficiently sophisticated to be able to handle any text input, irrespective of language style and register. However, due to the inherent complexity of the task, most parsing systems often fail to parse more casual texts accurately. In fact, they often encounter difficulty in parsing even formal text. We therefore need more reliable syntactic parsing of text irrespective of the register or style.

Link Grammar is a word based grammar ([1], [2]). It analyses sentences in terms of relationships between pairs of words. Historically, Link Parser is optimized for formal language and it often does not parse more casual text and sometimes even formal text. For instance, the parser is not able to parse phenomena like topicalization, extraposition, certain discourse elements (or sentence connectives), certain co-ordinate constructions, some indirect questions and other miscellaneous patterns, which were obtained after experiments on a large subset of a corpus.

This scope for improving the parser performance is the motivation for this paper. Although there are some other dependency parsers which are relatively more robust ([3], [4], [5], [6]), Link Parser is still in use, most notably for grammar checking in the word processor called AbiWord. Also, because of its rule based nature, it allows linguistically aware but computationally inexperienced users to increase the coverage of the parser. In our approach we do this without disturbing the core grammar. This decision is based on the realization that if the user is not very familiar with the intricacies of grammar (sometimes even if he is), it is possible that fixing one problem may create another problem. Hence, instead of disturbing the grammar, we introduce a preprocessor and a postprocessor, which are used to allow the parser to handle those sentences which are not handled by the original parser.

We first discuss how various constructions are not handled by the parser. Then we show how these constructions can be parsed after the addition of the preprocessor and the postprocessor. Section 2 is an introduction to the Link Parser. Section 3 presents a brief survey of the literature. Section 4 presents constructions that are not handled by the parser. In section 5, we introduce the preprocessor and the postprocessor and their role in the parser. In section 6, we present a summary of the results.

II. Link Parser

The Link Parser [1] is a syntactic parser, based on Link Grammar, in which every word is associated with a set of links, which have some associated suffixes to furnish various linguistic information like number etc. describing some properties of the word. In this kind of parsing, instead of constructing constituents in a tree like hierarchy, the parser basically performs analysis in terms of the relationship between pairs of words. Given a sentence, the parser assigns to it a syntactic structure, which consists of a set of labeled links connecting pairs of words. The link grammar is very closely related to dependency grammars that were formally expressed by Gaifman in 1965 ([7], [3], [1]). It uses knowledge about capitalization, numerical expressions, and a variety of punctuation symbols[1]. Figure 1 shows Link Grammar's output for a sample sentence.

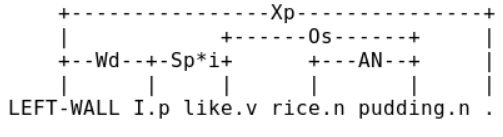

Fig. 1. Output of the Link Parser

In Figure 1 we see that **I** is linked on the right to the verb **like** with a link labeled *Sp*i*,[2] denoting Subject. (One can

[1]For more details see http://www.link.cs.cmu.edu/link/dict/

[2]Small letters represent gender, number, person etc.

also say that **like** is linked on the left to **I**). Similarly, **rice** is linked on the right to the noun **pudding** with link labeled *AN* denoting noun modifier to noun. And **rice** is linked on the left to the verb **like** with the link labeled *Os*, denoting Object. And **I** is linked on the left to the **LEFT-WALL**[3] with the link labelled *Wd*.

The Link Parser also gives the phrase or the constituent-structure representation of a sentence. These structures are derived on the basis of the Link Grammar links. For the sentence shown in Figure 1, the constituent structure is shown in Figure 2.

```
(S (NP I)
   (VP like
       (NP rice pudding))
   .)
```

Fig. 2. Link Parser's constituent analysis

III. Background

In this section we briefly present some background information on the Link Parser.

A. A Small Note on Link Grammar

The analysis given by the Link Parser is not always easy to interpret. Some of the labels are unlikely to be expected intuitively. To properly interpret we need to understand the entire system. This requires more effort from the user. Superficially, the analysis sometimes looks wrong even when it is not. For example, consider the parser's analysis of the sentence shown in Figure 3.

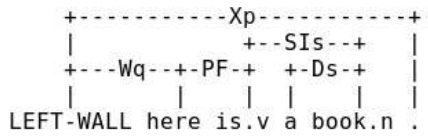

Fig. 3. An example of Link Parser's seemingly wrong analysis - 1

The parser gives a *Wq* label between **here** and *LEFT-WALL*. Note that *q* in *Wq* may be misleading as *q* is referring intuitively to question type label. Question type words, in fact, can get such a label, e.g. in Figure 4 *Whom* got a *Wq* label. In such a situation, one might wonder how can **here** be a question type word. The situation becomes more complex as even a preposition can have this label. See Figure 5 where **Among** gets a *Wq* link. To summarize, the same label can be assigned to different categories. To interpret the link labels, one has to understand the entire system. There is subject verb inversion (*SIs*) relation between **is** and **book** (in Figure 3) as given by the parser, although it is a declarative sentence. The meaning, according to parser's analysis, is that "SI" is used when the positions of the subject and verb are inverted, i.e., the verb comes first. That's the case here ("is" comes before "a book"). This inversion is different from the other subject-verb inversion like **Will you go to the school?**. In the latter case, **will** and **you** have been inverted.

This is just a small illustration of the complexity of the system from a user's point of view.

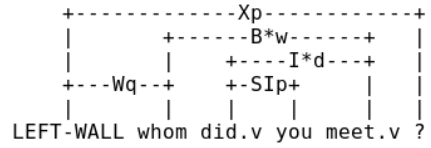

Fig. 4. Parser's analysis of a question sentence

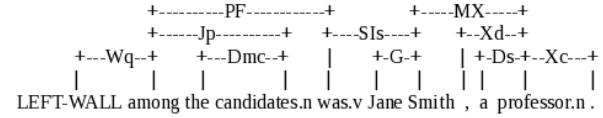

Fig. 5. Parser's seemingly wrong analysis - 2

B. Making Changes in the Grammar

Having illustrated the complexity of the Link Grammar we now propose that making changes inside the grammar is not an easy task for a common user. The Link Parser is now being maintained along with AbiWord, an open source word processor, which uses Link Grammar for grammar checking. Many changes and modifications are going on for improving the Link Parser. Understandably, since they are familiar with the grammar, their approach is to go inside the grammar and relaxing some conditions or making some other changes to improve the performance[4]. But this approach does not seem to work well because if we try to fix one problem by changing the grammar, it is very likely to create another problem somewhere else. This is the reason we chose not to disturb the existing grammar but to have an additional layer on top of it. We have some empirical evidence in support of this method. When we experimented on a corpus consisting of 2416 sentences on the recent version of the Link Parser and compared it with the old Link Parser[5]. The old Link Parser seems to show more consistent and better results than the new Link Parser. Details can be seen in Table I.

[3]The LEFT WALL is automatically inserted at the beginning of every sentence

[4]For more details http://www.abisource.com/projects/link-grammar/\#download

[5]When we say the Link Parser or the old Link Parser we mean the CMU 4.41b version. For more details http://www.link.cs.cmu.edu/link/ftp-site/link-grammar/link-4.1b/. And when we say the new Link Parser, we mean the version link-grammar-4.4.3, developed by the AbiWord group from the old version. For more details see http://www.abisource.com/downloads/link-grammar/.

	Handled	Unhandled	Coverage
Old version	1406	1010	58%
New version	1026	1390	42%

TABLE I
COMPARISON OF THE TWO VERSIONS

IV. CONSTRUCTIONS NOT HANDLED BY THE PARSER

In this section we discuss constructions which the parser fails to parse. When a word in a sentence is not recognized by the parser, it gives a message saying "No complete linkages found" and the marked word is shown in square brackets. We tried to generalize the patterns that are not handled by the parser such as extraposition [8], *if-then* clauses (when the *then*-clause has a missed subject), preposition stranding[6], topicalization, sentences having openers like *well*, *yes* etc., *what if* conditionals, some indirect questions, *think-so* constructions and other miscellaneous patterns. We also tried to classify the unhandled constructions that are similar in nature. For instance if we take the *what if* conditionals, sentences having openers like *yes* and *well* will be classified into one group. The reason for grouping them together is they all generally occur in the beginning of the sentence and since the Link Grammar is word based [9], it is important to keep every word in its most probable position. When we want to provide them the grammar, they all require some link to their right[7]. Some of the reasons for the parser not handling these constructions are :

- Special constructions
- The grammar may not be sufficient
- Conversational speech
- Lexicon might not be covered in the dictionary

We give some of the pattern frequencies[8] from the corpus in Table II. As we can observe from this table[9], it is clear that they are frequent enough. That the parser has failed to parse all of them shows the need for improvement.

A. Preposition Stranding

The parser is not able to parse a sentence when it ends with *from* and when it's object is *where*. From Figure 6 we can clearly see that not the whole of the sentence has been parsed. It is not the case that the parser does not handle all preposition stranding constructions in which a preposition with an object occurs somewhere other than immediately next to its object. For instance it can handle similar constructions with other stranded prepositions like *Whom did you give the book to?*

[6]Only when the sentence ends with *from* and if its object is *where*.

[7]For more details see http://www.link.cs.cmu.edu/link/dict/

[8]These frequencies are from the Corpus of Contemporary American English.

[9]Note: There were some other patterns which were sometimes handled by the parser and which we have not included in the list.

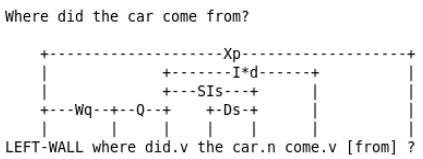

Fig. 6. Parser's analysis for preposition stranding

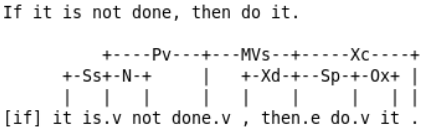

Fig. 7. Link Parser's analysis for the *if-then* clause

B. If-Then Clauses

The parser fails to parse the *if-then* clauses when the *then* clause subject is dropped. If we see Figure 7, it is clear that the parser was not able not handle it as it ignored the word *if*.

C. What-If Conditionals

The parser is also unable to handle the *what if* conditional clauses. If we look at Figure 8, it is obvious that the sentence was not fully parsed as the parser ignored both *what* and *if*.

D. Discourse Connectives or Sentence Openers

In the conversational text or even in some of the formal text, we often find many discourse connectives ([10], [11]) at the beginning of a sentence. Sample frequencies of some such connectives can be seen in Table II. When these sentences are passed through the Link Parser, it very often fails to parse them. For example sentences like *Well it's the end of time*. If

Pattern	Frequency
Well	251387
No	85376
Yes	77573
Here's	26957
Figure 1,2 ..8	19397
What if	10150
think so	9590
such that	2370
Ladies gentlemen	2248
idea how	1685
face to face	1530
of even	1345
of conscience	854
peace of mind	716
guess so	700
In conclusion	587
frame of mind	427
as you like	376
speak so	298
But why not	232
walk so	113

TABLE II
FREQUENCY DISTRIBUTION OF UNPARSED PATTERNS

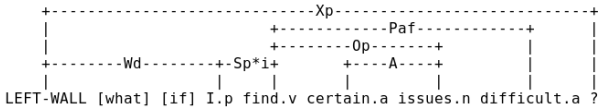

Fig. 8. Link Parser's analysis for the *what if* clause

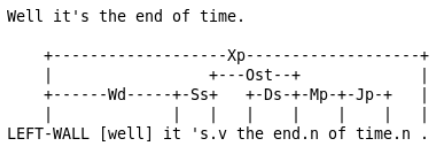

Fig. 9. Link Parser's analysis of discourse connectives - 1

we take a look at Figure 9, we can see that the parser has ignored the word *well* and performed the analysis for rest of the sentence. There is a similar problem with the sentence *Yes, my personal request has not been met* (See Figure 10). It ignores the word *Yes* and assigns structure to the rest of the sentence.

E. Extraposition

The sentence *Details have emerged of a secret plan to finance the rebels.* has not been parsed by the parser (See Figure 11). We can see that the parser did not parse *emerged* and *of* but interestingly handles the sentence when the noun is in its base position like *Details of a secret plan to finance the rebels have emerged*.

F. Such-That Pattern

The parser does not handle the sentences that have words *such that* (See Figure 12).

G. Topicalization

Topicalization is a phenomenon in which the focused expression appears in the sentence initial position. For example if we take the Figure 1 sentence, *Rice pudding* can be topicalized in the following manner: *Rice pudding I like.* The parser is not able to parse such a sentence.

H. Miscellaneous Patterns

The parser failed to parse the sentence *If an experiment disagrees with the current theory, the theory has to be changed, not the experiment*. We see in Figure 13 that the sentence was not fully parsed.

Similarly the sentence *If the House agrees, I shall do as Mr Evans has suggested.* was not parsed (See Figure 14).

Even when the preposition *of* is preceded by *conscience* the parser failed to parse (See Figure 15).

Still another construction is of the kind *That John came to the party surprised Mary* (See Figure 16).

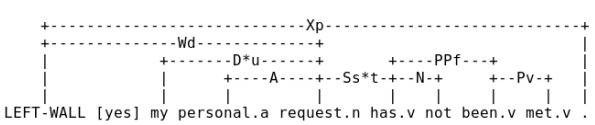

Fig. 10. Link Parser's analysis of discourse connectives - 2

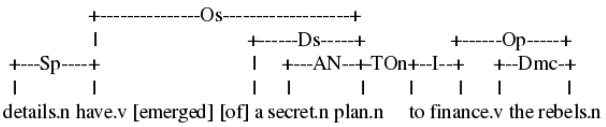

Fig. 11. Link parser' analysis for extraposition

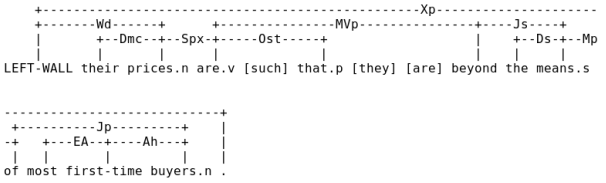

Fig. 12. Link Parser's analysis of *such that* construction

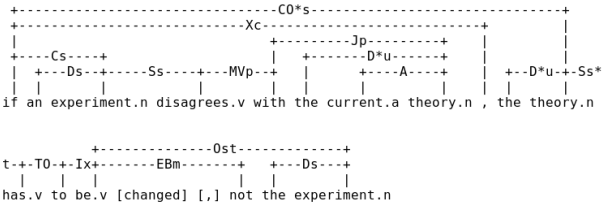

Fig. 13. Link parser's analysis of miscellaneous patterns - 1

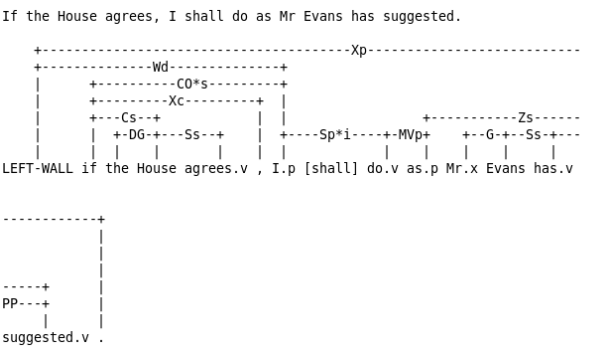

Fig. 14. Link Parser's analysis of miscellaneous patterns - 2

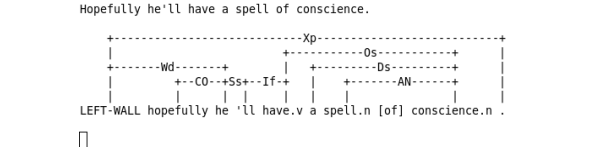

Fig. 15. Link parser's analysis of miscellaneous patterns - 3

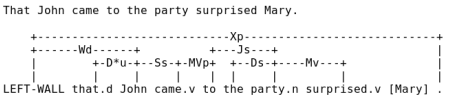

Fig. 16. Link parser's analysis of miscellaneous patterns - 4

V. Our Approach

While discussing unhandled constructions we have also shown in section 2 that relaxing or changing the grammar may not be easy for the problematic sentences to be parsed. Instead it will actually decrease its coverage because unexpected effects occur elsewhere. Hence, we propose an approach which will not disturb the existing grammar but will still be an addition to grammar.

A. Preprocessor

The reason for introducing the preprocessor is to handle the cases that are not parsed by the parser. For instance if we take the construction types from 4.2 to 4.4 all of them are absolutely grammatical but the link grammar is not able to handle them. Hence the parser reports 'No complete linkages found'. However, in all the cases the parser failed to connect the sentence opener with the other words. They have the similar structures, i.e., the ignored words are occurring at the sentence initial position. One way is to consider all those sentences as two 'sentences', for which we have a rule like **If the string begins with a 'Well', 'Yes', 'No', 'In conclusion' etc., split them into into two 'sentences'**[10]. The first 'sentence' will be kept aside and the other will be given as an input to the parser. Accordingly, *if-then* clause type (section 4.2), *what if* types (section 4.3) and discourse connectives (section 4.4) will get split into two sentences. In the case of *if then* clauses *if* will be one sentence and the rest (e.g., *it is not done, then do it.*) as another sentence. In the case of *what if*, *what if* will be one sentence and the rest (*I find certain issues difficult.*) will be the other. In the case of discourse elements, in a sentence like *Well, it's the end of time.*, the split will be as *Well* as one part and *it's the end of time* as the other part. After splitting the sentences, we will have the parse for Figures 8 as given in Figure 17. Once we get the parse of the sentence, we then add the first sentence to the parsed sentence in the postprocessing phase. Thus, the resulting parse will be as in Figure 18.

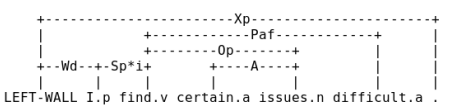

Fig. 17. Splitting sentences with *what if* conditionals

[10]Likewise, we list out all the possible discourse connectives ([10], [11]) that are present in the English language. If none of these are available, the preprocessor will not split the sentence.

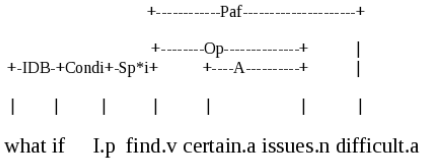

Fig. 18. Full parsed tree of the split sentence

Similarly, other discourse connectives like *in conclusion*, *yes* and so on will be handled with the split operation through preprocessing and postprocessing stages.

B. Postprocessor

The postprocessor that we use is similar to Link Grammar' dictionary[11]. It can be seen as a tiny dictionary where we provide labels to the sentences that are not handled by the parser, based on the rules that exist in the preprocessor. It will have the words that do not exist in the original dictionary and for which grammar would be provided to get the analysis. This tiny dictionary does not conflict with the original dictionary grammar. Writing grammar here is easier as we do not need to understand the entire system of Link Grammar. Writing an extension to the grammar outside the main dictionary[12] will not cause conflicts with the existing grammar (see Table III). In the postprocessor, the sentences that were splitted in the preprocessor phase will be joined in by the postprocessor along with link labels. For instance *what if* will be included in the dictionary that will be given a *Condi+* link to the right indicated with +, and the sentences that are having discourse connective will be given an X- link to the left after the splitting operation, where X means a variable. In the case of Condi+, the X- will become Condi- link.

Words	formula
What_if	Condi+
Well Yes No	CO+
Face_to_face	A+
Such_that	-Pa&Ce+

TABLE III
SOME ENTRIES IN THE POSTPROCESSING DICTIONARY

VI. Experiments and Results

One problem that we faced was that no gold standard is available for proper evaluation. And our purpose is not to compare with other grammars but to improve its coverage without disturbing the core grammar. This section describes the results of an experiment in which we applied our preprocessor to a randomly chosen subset of the WSJ corpus section 23. For these experiments, we used Kakkonen's evaluation tool ([12], [13]). Our purpose in doing this experiment was two fold.

[11]A dictionary consists of words and grammar that will have rules about how one word can be connected to the other words

[12]Note that it is a word based grammar

Sentences	2416
Total parses	1383369261
Parses/sentences	572586
Sentences with a complete parse	1404
An incomplete parse	909
No parse	103
Crashes	1
An unique parse	0
A complete unique parse	0
Panics	55
Coverage	0.5808854

TABLE IV
LINK PARSER COVERAGE EVALUATION RESULTS WITHOUT PRE- AND POSTPROCESSOR

Sentences	2416
Total parses	1344183665
Parses/sentences	556367
Sentences with a complete parse	1756
An incomplete parse	561
No parse	99
Crashes	1
An unique parse	0
A complete unique parse	0
Panics	31
Coverage	0.7265205

TABLE V
LINK PARSER COVERAGE EVALUATION RESULTS WITH PRE- AND POSTPROCESSOR

We wanted to determine to what extent our preprocessor and postprocessor actually increase the converge of the parser by handling the problematic constructions. We also wanted to see whether our approach allows the end user to contribute to the development of the parsing system to make it more robust. Table IV gives the results without the preprocessor and the postprocessor, while Table V gives the results with them. As the table shows, there was a significant increase (14%) in the coverage of the parser by using our simple approach. And we were able to achieve this relatively easily without causing any adverse effects.

There are still many issues that can cause problems. For example, we can take care of the topicalization constructions, but the problem is that it is hard to automatically identify the cases of topicalization. Similar is the case with extraposition.

VII. CONCLUSION

In this paper we have proposed an approach to improve the Link Parser's coverage. The proposed approach has not only shown significant improvement but also allows further development by users not very familiar with the internals of the grammar. We first discussed various syntactic constructions that have not been handled by the parser and their frequency distribution in a representative corpus. In order to handle these constructions we introduced pre and postprocessors. The former identifies the cases where the parser fails to parse, while the latter provide the grammar to add link labels in such cases. We also showed why the obvious approach (increasing the grammar using additional rules or changing the grammar) does not easily work. Our approach allows a common user to contribute to the development of the parser with little knowledge of Link Grammar. Also, even though good statistical parsers may be available, there are applications like grammar checking where rule based parsers like the Link Parser may have an edge.

ACKNOWLEDGEMENT

Authors would like to acknowledge the guidance of Dr. Vineet Chaitanya whose suggestion originated this work. We convey our gratitude to him for his guidance, support and direction, especially for reading the drafts and giving feedback. Authors also sincerely thank Prof. Aditi Mukherjee of Osmania University for proof reading the paper.

REFERENCES

[1] D. Sleator and D. Temperley, "Parsing English with a link grammar," *Arxiv preprint cmp-lg/9508004*, 1995.
[2] D. Grinberg, J. Lafferty, and D. Sleator, "A robust parsing algorithm for link grammars," *Arxiv preprint cmp-lg/9508003*, 1995.
[3] G. Schneider, "A linguistic comparison of constituency, dependency and link grammar," *Zurich, Switzerland: Licentiate Thesis, University of Zurich*, 1998.
[4] Collins, M. J., "Head-driven statistical models for natural language parsing," in *PhD thesis, University of Pennsylvania*, 1999.
[5] M. Collins and N. Duffy, "Discriminative reranking for natural language parsing," *Computational Linguistics*, 2005.
[6] J. Hall, J. Nivre, and J. Nilsson, "Discriminative classifiers for deterministic dependency parsing," in *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 2006, pp. 316–323.
[7] H. Gaifman, "Dependency systems and phrase-structure systems," *Information and Control*, pp. 304–337, 1965.
[8] J. Ouhalla, *Introducing transformational grammar: from Principles and Parameters to Minimalism*. Hodder Arnold, 1999.
[9] R. Hudson, *Word grammar*. Blackwell, 1984.
[10] B. Webber, A. Knott, and A. Joshi, "Multiple discourse connectives in a lexicalized grammar for discourse," *STUDIES IN LINGUISTICS AND PHILOSOPHY*, pp. 229–246, 2001.
[11] E. Miltsakaki, R. Prasad, A. Joshi, and B. Webber, "Annotating discourse connectives and their arguments," in *Proceedings of the HLT/NAACL Workshop on Frontiers in Corpus Annotation*, 2004, pp. 9–16.
[12] T. Kakkonen, "Framework and resources for natural language parser evaluation," Ph.D. dissertation, Department of Computer Science and Statistics, University of Joensuu.
[13] ——, "Robustness evaluation of two ccg, a pcfg and a link grammar parsers," *Arxiv preprint arXiv:0801.3817*, 2008.