The Stanford typed dependencies representation

Marie-Catherine de Marneffe
Linguistics Department
Stanford University
Stanford, CA 94305
mcdm@stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305
manning@stanford.edu

Abstract

This paper examines the Stanford typed dependencies representation, which was designed to provide a straightforward description of grammatical relations for any user who could benefit from automatic text understanding. For such purposes, we argue that dependency schemes must follow a simple design and provide semantically contentful information, as well as offer an automatic procedure to extract the relations. We consider the underlying design principles of the Stanford scheme from this perspective, and compare it to the GR and PARC representations. Finally, we address the question of the suitability of the Stanford scheme for parser evaluation.

1 Introduction

The Stanford typed dependencies representation was designed to provide a simple description of the grammatical relationships in a sentence that could easily be understood and effectively used by people without linguistic expertise who wanted to extract textual relations. The representation was not designed for the purpose of parser evaluation. Nevertheless, we agree with the widespread sentiment that dependency-based evaluation of parsers avoids many of the problems of the traditional Parseval measures (Black et al., 1991), and to the extent that the Stanford dependency representation is an effective representation for the tasks envisioned, it is perhaps closer to an appropriate task-based evaluation than some of the alternative dependency representations available. In this paper we examine the representation and its underlying design principles, look at how this representation compares with other dependency representations in ways that reflect the design principles, and consider its suitability for parser evaluation.

A major problem for the natural language processing (NLP) community is how to make the very impressive and practical technology which has been developed over the last two decades approachable to and usable by everyone who has text understanding needs. That is, usable not only by computational linguists, but also by the computer science community more generally and by all sorts of information professionals including biologists, medical researchers, political scientists, law firms, business and market analysts, etc. Thinking about this issue, we were struck by two facts. First, we noted how frequently WordNet (Fellbaum, 1998) gets used compared to other resources, such as FrameNet (Fillmore et al., 2003) or the Penn Treebank (Marcus et al., 1993). We believe that much of the explanation for this fact lies in the difference of complexity of the representation used by the resources. It is easy for users not necessarily versed in linguistics to see how to use and to get value from the straightforward structure of WordNet. Second, we noted the widespread use of MiniPar (Lin, 1998) and the Link Parser (Sleator and Temperley, 1993). This clearly shows that (i) it is very easy for a non-linguist thinking in relation extraction terms to see how to make use of a dependency representation (whereas a phrase structure representation seems much more foreign and forbidding), and (ii) the availability of high quality, easy-to-use (and preferably free) tools is essential for driving broader use of NLP tools.[1]

[1]On the other hand, evaluation seems less important; to the best of our knowledge there has never been a convincing and thorough evaluation of either MiniPar or the Link Grammar

This paper advocates for the Stanford typed dependencies representation (henceforth SD) being a promising vehicle for bringing the breakthroughs of the last 15 years of parsing research to this broad potential user community. The representation aims to provide a simple, habitable design. All information is represented as binary relations. This maps straightforwardly on to common representations of potential users, including the logic forms of Moldovan and Rus (Moldovan and Rus, 2001),[2] semantic web Resource Description Framework (RDF) triples (http://www.w3.org/RDF/), and graph representations (with labeled edges and nodes). Unlike many linguistic formalisms, excessive detail is viewed as a defect: information that users do not understand or wish to process detracts from uptake and usability. The user-centered design process saw the key goal as representing semantically contentful relations suitable for relation extraction and more general information extraction uses. The design supports this use by favoring relations between content words, by maintaining semantically useful closed class word information while ignoring linguistic decisions less relevant to users, and by not representing less used material about linguistic features such as tense and agreement. The SD scheme thus provides a semantic representation simple and natural enough for people who are not (computational) linguists but can benefit from NLP tools.

2 Design choices and their implications

2.1 Design principles

The style of the SD representation bears a strong intellectual debt to the framework of Lexical-Functional Grammar (Bresnan, 2001), and, more directly, it owes a debt to both the sets of grammatical relations and the naming defined in two representations that follow an LFG style: the GR (Carroll et al., 1999) and PARC (King et al., 2003) schemes. These were used as a starting point for developing the Stanford dependencies (de Marneffe et al., 2006). But where the SD scheme deviates from GR, PARC, and its LFG roots is that it has been designed to be a practical model of sentence representation, particularly in the context of relation extraction tasks.

parser.

[2]The logic forms of Moldovan and Rus are in the form of a predicate calculus representation, although not one that represents such things as operator scope in a way that most would expect of a predicate calculus representation.

SD makes available two options, suited to different use cases: in one, every word of the original sentence is present as a node with relations between it and other nodes, whereas in the latter, certain words are "collapsed" out of the representation, making such changes as turning prepositions into relations. The former is useful when a close parallelism to the source text words must be maintained, whereas the latter is intended to be more useful for relation extraction and shallow language understanding tasks. Here, we discuss only the latter representation; see (de Marneffe et al., 2006) for a discussion of both options and the precise relationship between them.

The intended use cases of usability by people who are not (computational) linguists and suitability for relation extraction applications led SD to try to adhere to the following design principles (DPs):

1. Everything is represented uniformly as some binary relation between two sentence words.
2. Relations should be semantically contentful and useful to applications.
3. Where possible, relations should use notions of traditional grammar for easier comprehension by users.
4. Underspecified relations should be available to deal with the complexities of real text.
5. Where possible, relations should be between content words, not indirectly mediated via function words.
6. The representation should be spartan rather than overwhelming with linguistic details.

We illustrate many of them in the rest of this section, using example sentences which were made available for the Parser Evaluation Shared Task.

The grammatical relations of SD are arranged in a hierarchy, rooted with the most generic relation, *dependent*. The hierarchy contains 56 grammatical relations. When the relation between a head and its dependent can be identified more precisely, relations further down in the hierarchy are used, but when it is unclear, more generic dependencies are possible (DP1, DP4). For example, the *dependent* relation can be specialized to *aux* (auxiliary), *arg* (argument), or *mod* (modifier). The *arg* relation is further divided into the *subj* (subject) relation and the *comp* (complement) relation, and so on. The backbone of this hierarchy is quite similar to that in GR, but there are some crucial differences.

2.2 Comparison with GR and PARC

The SD scheme is not concerned with the argument/adjunct distinction which is largely useless in practice. In contrast, NP-internal relations are an inherent part of corpus texts and are critical in real-world applications. The SD scheme therefore includes many relations of this kind: *appos* (appositive modifier), *nn* (noun compound), *num* (numeric modifier), *number* (element of compound number) and *abbrev* (abbreviation), etc. (DP2). For instance, in the sentence *"I feel like a little kid," says a gleeful Alex de Castro, a car salesman, who has stopped by a workout of the Suns to slip six Campaneris cards to the Great Man Himself to be autographed* (*WSJ-R*), we obtain the following relations under the SD representation:

SD	appos(Castro, salesman)
	num(cards, six)
	nn(cards, Campaneris)

The numeric modifier relation between *cards* and *six* is also standard in the PARC and GR schemes. PARC provides an apposition relation between *salesman* and *Alex de Castro*, whereas GR only identifies *salesman* as a text adjunct of *Castro*. But on the whole, SD makes more fine-grained distinctions in the relations, which are needed in practice. The *adjunct* dependency of the PARC scheme lumps together different relations. For example, the adjectival modifier *gleeful* in the sentence above will not be marked distinctively from the preposition modifying *workout*, nor from the relation between the verbs *stop* and *slip*:

PARC	adjunct(Alex de Castro, gleeful)
	adjunct(kid, little)
	adjunct(stop, slip)
	adjunct(workout, of)

The SD output for the relations between these words looks as follows:

SD	amod(Castro, gleeful)
	amod(kid, little)
	xcomp(stop, slip)
	prep_of(workout, Suns)

The comparison between the two outputs shows that SD proposes a larger set of dependencies, capturing relation differences which can play a role in applications (DP2), while sticking to notions of traditional grammar (DP3).

The SD scheme also chooses content words as heads of the dependencies (DP5). Auxiliaries, complementizers, and so on, are dependents of them. This choice in design is driven by the kind of information that is useful for applications. For instance, in the sentence *Considered as a whole, Mr. Lane said, the filings required under the proposed rules "will be at least as effective, if not more so, for investors following transactions" (WSJ-R)*, *effective* is chosen as the head of the quoted phrase. This enables the representation to have a direct dependency (*nsubj* for nominal subject) between the key content words *effective* and *filings*. Such a link is more difficult to infer from the GR scheme, where *be* is chosen as the head. However the relation between *effective* and *filings* is key to extracting the gist of the sentence semantics, and it is therefore important for applications to be able to retrieve it easily. Also, in the case of structures involving copular verbs, a direct link between the subject and the complement enables equivalent representations across languages (in Chinese, for example, copulas are not explicitly expressed). Such parallel representations should presumably help machine translation, and this was a further motivation for choosing content words as heads.

Another instance where direct links between content words is useful is the case of prepositional complements. The SD scheme offers the option of "collapsing" dependencies involving a preposition (DP5). In the example above, instead of having two relations *adjunct*(workout, of) and *obj*(of, Suns) as in PARC or *ncmod*(workout, of) and *dobj*(of, Suns) as in GR, SD provides a direct relation between the content words: *prep_of*(workout, Suns). Prepositions often work as role markers, and this type of link facilitates the extraction of how the two content words are related; and thus these links are often used by downstream applications (Lin and Pantel, 2001; Snow et al., 2005). The usefulness of the representation is exemplified in the sentence *A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice (WSJ-R)* for which SD gives direct links between the entities joined through the preposition *such as*:

SD	prep_such_as(crops, cotton)
	prep_such_as(crops, soybeans)
	prep_such_as(crops, rice)

A similar collapsing treatment takes place for conjuncts (DP5). Consider the following sentence: *Bell, based in Los Angeles, makes and distributes*

```
SD   nsubj(makes-8, Bell-1)
     nsubj(distributes-10, Bell-1)
     partmod(Bell-1, based-3)
     nn(Angeles-6, Los-5)
     prep_in(based-3, Angeles-6)
     conj_and(makes-8, distributes-10)
     amod(products-16, electronic-11)
     conj_and(electronic-11, computer-13)
     amod(products-16, computer-13)
     conj_and(electronic-11, building-15)
     amod(products-16, building-15)
     dobj(makes-8, products-16)
```

Figure 1: SD representation for *Bell, based in Los Angeles, makes and distributes electronic, computer and building products.*

electronic, computer and building products (WSJ-R). Figures 1 and 2 give the full dependency output from SD and GR, respectively. The numbers after the words in the SD representation indicate the word position in the sentence.[3] From the SD representation, one can easily see that the sentence talks about *electronic products* and *computer products* as well as *building products*. By collapsing the dependencies involving conjuncts, the output produced is closer to the semantics of the sentence, and this facilitates information extraction (DP2). This information is not straightforwardly apparent in the GR scheme (see figure 2), nor in the PARC scheme which follows a similar treatment of conjuncts.

Another choice in the design has been to consistently have binary relations (DP1). All the dependencies form a triple: a grammatical relation holding between two words (head and dependent). This gives uniformity to the representation and renders it very readable, critical features for a user-centered design. Furthermore, all the information can be represented by a directed graph, enabling the creation of both a limpid visual representation for humans and a canonical data structure for software. Moreover, it maps straightforwardly on to semantic web representations such as OWL and RDF triples, as exploited in (Zouaq et al., 2006; Zouaq et al., 2007).

This design choice limits the kind of information offered by the SD scheme. For instance, the PARC scheme contains much more information

[3]Without word position, the representation is deficient if the same word occurs more than once in a sentence.

```
GR   (passive based)
     (ncsubj based Bell obj)
     (ta bal Bell based)
     (iobj _ based in)
     (dobj in Angeles)
     (ncmod _ Angeles Los)
     (conj and makes)
     (conj and distributes)
     (conj and electronic)
     (conj and computer)
     (conj and building)
     (ncsubj and Bell _)
     (dobj and products)
     (ncmod _ products and)
```

Figure 2: GR representation for *Bell, based in Los Angeles, makes and distributes electronic, computer and building products.*

about individual words, such as verb tense and aspect, noun number and person, type of NE for proper nouns, pronoun form, adjective degree, etc. For the sentence in figures 1 and 2, the following information is available for the word *Los Angeles* in the PARC scheme:

```
PARC   num(Los Angeles∼5, sg)
       pers(Los Angeles∼5, 3)
       proper(Los Angeles∼5, location)
```

This kind of information is indubitably valuable, but is often less used in practice, and does not per se pertain to dependency data. Adding it lengthens an output already complex enough, and impedes readability and convenience. Thus, SD does not provide such overwhelming detail (DP6).

2.3 Trading off linguistic fidelity and usability

We feel that turning prepositions into relations is useful for 98% of users 98% of the time. Nevertheless opting for usability in this way causes the SD scheme to sacrifice some linguistic fidelity. One instance is that modifiers of prepositions are dependent on the verb (or more precisely, on the head of the clause in which they appear) and not on the preposition itself. In *Bill went over the river and right through the woods*, *right* will be an adverbial modifier of *went*. In *He had laughed, simultaneously mocking the stupidity of government by cosmetics and confessing that he was also a part of it, just as he was part of government by voice coach and acting coach (BNC)*, *just* which modifies *as* will be a dependent of the head of the adverbial clause, i.e., *part*. This induces some distortion in the exact semantics of the sentence.

The interaction between preposition collapsing and PP conjunction is another instance in which the SD treatment slightly alters the semantics of the sentence. Consider again the sentence *Bill went over the river and right through the woods*. Both prepositions, *over* and *through*, are governed by the verb *went*. To avoid disjoint subgraphs when collapsing the relations, examples like this are transformed into VP coordination, which requires making a copy of the word *went*. This gives the following representation, which corresponds to a sentence like *Bill went over the river and went right through the woods*:

SD prep_over(went-2, river-5)
prep_through(went-2', woods-10)
conj_and(went-2, went-2')

Not collapsing the relations in such a case would prevent the alteration of the semantics, but would lead to a non-uniform treatment of prepositions. Uniformity is key for readability and user convenience. It seems therefore reasonable to use a representation which sacrifices the exact semantics of the original sentence by producing a sentence roughly equivalent, but which ensures uniformity across relations.

3 The formalism and the tool

Two vital conditions for the success of a dependency scheme are to provide a suitable representation for users as well as a tool that is easy to use. Sagae et al. (2008) note that the availability of an automatic procedure to convert phrase structure parses to SD is the reason for its use in evaluations of parsers in the biomedical domain. The primary focus of the SD scheme, however, has been to offer grammatical relations appropriate for end-users.

The Stanford parser[4] comes with a tool, described in (de Marneffe et al., 2006), which provides for the rapid extraction of the grammatical relations from phrase structure parses. Structural configurations are used to define grammatical roles: the semantic head of each constituent of the parse is identified, using rules akin to the Collins head rules, but modified to retrieve the semantic head of the constituent rather than the syntactic head. As mentioned, content words are chosen as heads, and all the other words in the constituent depend on this head. To retrieve adequate heads from a semantic point of view, heuristics are used to inject more structure when the Penn Treebank gives only flat constituents, as is often the case for conjuncts, e.g., (NP the new phone book and tour guide), and QP constituents, e.g., (QP more than 300). Then for each grammatical relation, patterns are defined over the phrase structure parse tree using the tree-expression syntax defined by tregex (Levy and Andrew, 2006). Conceptually, each pattern is matched against every tree node, and the matching pattern with the most specific grammatical relation is taken as the type of the dependency.

The automatic extraction of the relations is not infallible. For instance, in the sentence *Behind their perimeter walls lie freshly laundered flowers, verdant grass still sparkling from the last shower, yew hedges in an ecstasy of precision clipping (BNC)*, the system will erroneously retrieve apposition relations between *flowers* and *grass*, as well as between *flowers* and *hedges* whereas these should be *conj_and* relations. The system is clueless when there is no overt maker of conjunction.

Another limitation of the tool is the treatment of long-distance dependencies, such as *wh*-movement and control/raising: the system cannot handle long-distance dependencies that cross clauses. In a sentence like *What does he think?*, the system will correctly find that *what* is a direct object of *think*:

SD dobj(think-4, What-1)
aux(think-4, does-2)
nsubj(think-4, he-3)

However in a sentence such as *Who the hell does he think he's kidding? (BNC)*, the automatic extraction will fail to find that *who* is the direct object of *kidding*. Here, it is vital to distinguish between SD as a representation versus the extant conversion tool. Long-distance dependencies are not absent from the formalism, but the tool does not accurately deal with them.[5]

4 Stanford dependencies in practice

SD has been successfully used by researchers in different domains. In the PASCAL Recognizing

[4] http://nlp.stanford.edu/software/lex-parser.shtml

[5] As possible future work, we have thought of using a tool such as the one of Levy and Manning (2004) to correctly determine long distance dependencies, as input to the current dependency conversion system. This would presumably be effective, but would make the conversion process much heavier weight.

Textual Entailment (RTE) challenges (Dagan et al., 2006; Giampiccolo et al., 2007), the increase in the use of SD is clearly apparent. The goal in these challenges consists of identifying whether one sentence follows from a piece of text and general background knowledge, according to the intuitions of an intelligent human reader. In 2007, out of the 21 systems which participated in the challenge, 5 used the SD representation, whereas the year before only the Stanford entry was using it.

SD is also widely present in the bioinformatic world where it is used with success (Erkan et al., 2007; Greenwood and Stevenson, 2007; Urbain et al., 2007; Clegg, 2008). Fundel et al. (2007) found that, in extraction of relations between genes and proteins, a system based on the SD scheme greatly outperformed the previous best system on the LLL challenge dataset (by an 18% absolute improvement in F-measure). Airola et al. (2008) provide more systematic results on a number of protein-protein interaction datasets. Their graph kernel approach uses an all-dependency-paths kernel which allows their system to consider full dependency graphs. Their system is based on the SD scheme, and they demonstrate state-of-the-art performance for this approach.

In the biomedical domain, SD has recently been used in evaluations of parsers (Clegg and Shepherd, 2007; Pyysalo et al., 2007a). Pyysalo et al. (2007a) assessed the suitability of the SD scheme over the Link Grammar dependency scheme in an application-oriented evaluation. The Link Parser indeed uses a very fine-grained set of relations, which often makes distinctions of a structural rather than a semantic nature. One example is the MX relation which "connects modifying phrases with commas to preceding nouns ('The DOG, a POODLE, was black'; 'JOHN, IN a black suit, looked great')." The Link Parser uses a different set of dependency types for dependencies appearing in questions and relative clauses. Another example is the prepositional phrase where alternative attachment structures are indicated by different relations. Many of these distinctions are too fine and non-semantic to be of practical value. The SD scheme, by aiming for an intermediate level of granularity, and targeting semantic dependencies, provides a more adequate representation for applications. Therefore, to increase the usability of the BioInfer corpus (Pyysalo et al., 2007b), which provides manually annotated data for information extraction in the biomedical domain and originally followed the Link Grammar scheme, Pyysalo et al. (2007a) developed a version of the corpus annotated with the SD scheme. They also made available a program and conversion rules that they used to transform Link Grammar relations into SD graphs, which were then hand-corrected (Pyysalo et al., 2007b). While a limited amount of gold standard annotated data was prepared for the Parser Evaluation Shared Task, this is the main source of gold-standard SD data which is currently available.

In other domains, Zhuang et al. (2006) uses the representation to extract opinions about features in reviews and Meena and Prabhakar (2007) uses it to improve the quality of sentence-level sentiment analysis. The open information extraction system TEXTRUNNER (Banko et al., 2007) also makes use of the SD graph representation: its first module uses the Stanford parser and the dependency tool to automatically identify and label trustworthy and untrustworthy extractions. Even in theoretical linguistic work, SD has proven very useful: it has hugely facilitated data extraction from corpora, in the context of the NSF-funded project "Dynamics of probabilistic grammar" carried out at the Stanford Linguistics department.

5 Suitability for parser evaluation

When seeking a gold-standard dependency scheme for parser evaluation, the ultimate goal of such an evaluation is an important question. It is necessary to contrast the two different forms that evaluation can take: extrinsic task-based evaluation and intrinsic evaluation. We tend to agree with Mollá and Hutchinson (2003) that intrinsic evaluations have limited value and that task-based evaluation is the correct approach. Some of the results of the previous section at least broadly support the utility of the SD scheme for practical use in higher-level tasks. Nevertheless, given the current trend in the NLP community as well as in other fields such as bioinformatics, where the advantage of dependency representations for shallow text understanding tasks has become salient, we would argue, following Clegg and Shepherd (2007), that dependency-based evaluation is close to typical user tasks. Moreover, it avoids some of the known deficiencies of other parser evaluation measures such as Parseval (Carroll et al., 1999).

Recent work on parser evaluation using dependency graphs in the biomedical domain confirms that researchers regard dependency-based evaluation as a more useful surrogate for extrinsic task-based evaluation (Clegg and Shepherd, 2007; Pyysalo et al., 2007a). In their evaluation, Clegg and Shepherd (2007) aimed at analyzing the capabilities of syntactic parsers with respect to semantically important tasks crucial to biological information extraction systems. To do so, they used the SD scheme, which provides "a de facto standard for comparing a variety of constituent parsers and treebanks at the dependency level," and they assessed its suitability for evaluation. They found that the SD scheme better illuminates the performance differences between higher ranked parsers (e.g., Charniak-Lease parser (Lease and Charniak, 2005)), and lower ranked parsers (e.g., the Stanford parser (Klein and Manning, 2003)). Their parser evaluation accommodates user needs: they used the collapsed version of the dependency graphs offered by the SD scheme, arguing that this is the kind of graph one would find most useful in an information extraction project. Although Clegg and Shepherd (2007) also favor dependency graph representations for parser evaluation, they advocate retention of parse trees so information lost in the dependency structures can be accessed.

In essence, any existing dependency scheme could be adopted as the gold-standard for evaluation. However if one believes in ultimately valuing extrinsic task-based evaluation, a dependency representation which proposes a suitable design for users and user tasks is probably the best surrogate for intrinsic evaluation. Moreover, the existence of tools for automatically generating and converting dependency representations has aided greatly in making parser comparison possible across different formalisms. We believe that the SD scheme approaches these goals. If one accepts the goals set here, in order to enforce uniformity between application and evaluation, it seems sensible to have a unique scheme for both purposes. Some of the positive results from use of the SD representation, as well as the evaluations carried out in the biomedical field, point to the usability of the SD scheme for both purposes.

Acknowledgments

We wish to thank Andrew Brian Clegg and Sampo Pyysalo for their useful feedback on the dependency extraction tool. Their comments enabled us to improve the tool. We also thank the workshop reviewers for their helpful comments.

References

Airola, Antti, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. A graph kernel for protein-protein interaction extraction. In *Proceedings of BioNLP 2008: Current Trends in Biomedical Natural Language Processing (ACL08)*.

Banko, Michele, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*.

Black, E., S. Abney, D. Flickinger, 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 *Proceedings, Speech and Natural Language Workshop*, pages 306–311, Pacific Grove, CA. DARPA.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Blackwell, Oxford.

Carroll, John, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL workshop on Linguistically Interpreted Corpora (LINC)*.

Clegg, Andrew B. and Adrian J. Shepherd. 2007. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8:24.

Clegg, Andrew B. 2008. *Computational-Linguistic Approaches to Biological Text Mining*. Ph.D. thesis, School of Crystallography, Birkbeck, University of London.

Dagan, Ido, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In et al., Quinonero-Candela, editor, *MLCW 2005, LNAI Volume 3944*, pages 177–190. Springer-Verlag.

de Marneffe, Marie-Catherine, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-06*.

Erkan, Gunes, Arzucan Ozgur, and Dragomir R. Radev. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Fellbaum, Christiane. 1998. *WordNet: an electronic lexical database*. MIT Press.

Fillmore, Charles J., Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16:235–250.

Fundel, Katrin, Robert Küffner, and Ralf Zimmer. 2007. RelEx relation extraction using dependency parse trees. *Bioinformatics*, 23.

Giampiccolo, Danilo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9.

Greenwood, Mark A. and Mark Stevenson. 2007. A semi-supervised approach to learning relevant protein-protein interaction articles. In *Proceedings of the Second BioCreAtIvE Challenge Workshop, Madrid, Spain*.

King, Tracy H., Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald Kaplan. 2003. The PARC 700 dependency bank. In *4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

Lease, Matthew and Eugene Charniak. 2005. Parsing biomedical literature. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP'05)*.

Levy, Roger and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *LREC 2006*. http://www-nlp.stanford.edu/software/tregex.shtml.

Levy, Roger and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *ACL 42*, pages 328–335.

Lin, Dekang and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Lin, Dekang. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems, Granada, Spain*.

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

Meena, Arun and T. V. Prabhakar. 2007. Sentence level sentiment analysis in the presence of conjuncts using linguistic analysis. In *Advances in Information Retrieval*, volume 4425 of *Lecture Notes in Computer Science*. Springer.

Moldovan, Dan I. and Vasile Rus. 2001. Logic form transformation of wordnet and its applicability to question answering. In *Meeting of the Association for Computational Linguistics*, pages 394–401.

Mollá, Diego and Ben Hutchinson. 2003. Intrinsic versus extrinsic evaluations of parsing systems. In *Proceedings of the Workshop on Evaluation Initiatives in Natural Language Processing*, pages 43–50. European Association for Computational Linguistics.

Pyysalo, Sampo, Filip Ginter, Katri Haverinen, Juho Heimonen, Tapio Salakoski, and Veronika Laippala. 2007a. On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. In *Proceedings of BioNLP 2007: Biological, translational, and clinical language processing (ACL07)*.

Pyysalo, Sampo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007b. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50.

Sagae, Kenji, Yusuke Miyao, and Jun'ichi Tsujii. 2008. Challenges in mapping of syntactic representations for framework-independent parser evaluation. In *Proceedings of the Workshop on Automated Syntatic Annotations for Interoperable Language Resources at the First International Conference on Global Interoperability for Language Resources (ICGL'08)*.

Sleator, Daniel D. and Davy Temperley. 1993. Parsing English with a link grammar. In *Third International Workshop on Parsing Technologies*.

Snow, Rion, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of NIPS 2004*.

Urbain, Jay, Nazli Goharian, and Ophir Frieder. 2007. IIT TREC 2007 genomics track: Using concept-based semantics in context for genomics literature passage retrieval. In *The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings*.

Zhuang, Li, Feng Jing, Xiao yan Zhu, and Lei Zhang. 2006. Movie review mining and summarization. In *Proc. ACM Conference on Information and Knowledge Management (CIKM)*.

Zouaq, Amal, Roger Nkambou, and Claude Frasson. 2006. The knowledge puzzle: An integrated approach of intelligent tutoring systems and knowledge management. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006)*, pages 575–582.

Zouaq, Amal, Roger Nkambou, and Claude Frasson. 2007. Building domain ontologies from text for educational purposes. In *Proceedings of the Second European Conference on Technology Enhanced Learning: Creating new learning experiences on a global scale*.