Syntactic Representations Considered for Frame-semantic Analysis

Richard Johansson and Pierre Nugues
Lund University
Department of Computer Science

Abstract

We address the question of which syntactic representation is best suited for FrameNet-based semantic analysis of English text. We compare analyzers based on dependencies and constituents, and a dependency syntax with a rich set of grammatical functions with one with a smaller set. Our study demonstrates that dependency-based and constituent-based analyzers give roughly equivalent performance, and that a richer set of functions has a positive influence on argument classification for verbs.

1 Introduction

The role-semantic paradigm (see [12], *inter alia*) is a prominent model in automatic semantic analysis. With a few exceptions (e.g. [9]), role-based semantic analysis relies crucially [13, 21] on some sort of syntactic representation as input.

Since syntactic structure is generally considered important for semantic analysis, the question arises on how to best design this representation to facilitate analysis. Most systems for automatic role-semantic analysis have used Penn-style constituent syntax [15] via Collins' [8] or Charniak's [7] parser. An exception is a semantic analyzer based on Combinatorial Category Grammar [11]. Dependency syntax has rarely been used for this task, and with mediocre results [20], although many linguists argue that dependencies are a more intuitive way to explain the syntax-semantics interface [18].

On the other hand, dependency syntax and parsing have recently attracted much interest in the NLP research community [6], which has led to significant improvements in parsing accuracy. It is therefore interesting to investigate whether role-semantic analyzers based on the output of state-of-the-art dependency parsers can be competitive with analyzers based on constituent syntax. In addition, we can posit that the expressivity of the dependency structures (i.e. the size of the set of grammatical functions) can be an important parameter influencing the performance of the analysis.

In this paper, we describe a set of experiments in which we compare the performance of constituent-based and dependency-based semantic analyzers on the three main subtasks of FrameNet-based role-semantic analysis.

2 Overview of FrameNet

FrameNet [3] is a medium-sized lexical database that lists descriptions of English words in Fillmore's paradigm of Frame Semantics [10]. In this framework, the relations between predicates (or in FrameNet terminology, *target words*) and their arguments are described by means of *semantic frames*. A frame can intuitively be thought of as a template that defines a set of slots, the *frame elements*, that represent parts of a conceptual structure and correspond to its prototypical participants or properties. In Example 1, the target word *want* and its arguments form a compound structure by means of the frame DESIRING. This frame has 14 possible frame elements in total. Three of them are instantiated in the example below: EXPERIENCER, FOCAL_PARTICIPANT, and EVENT:

(1) Do [I]$_{\text{EXPERIENCER}}$ **want** [him]$_{\text{FOCAL\_PARTICIPANT}}$ [to see me]$_{\text{EVENT}}$?

The initial versions of FrameNet were restricted to describing situations and events, i.e. typically verbs and their nominalizations. Currently, however, FrameNet defines frames for a wider range of semantic relations and word classes. The set of frames for a word also serves as its sense inventory, which in most cases is more coarse-grained than its WordNet counterpart. For instance, for the verb *want*, the DESIRING frame roughly corresponds to WordNet senses 1–4, while the POSSESSION frame corresponds to WordNet sense 5.

The FrameNet package consists of the following main parts:

- A *frame ontology* consisting of a set of frames, frame elements for each frame, and relations such as *is-a*, *uses*, and *causative-of* between frames.
- A list of *lexical units*: word senses paired with their corresponding frames.
- A collection of *example sentences* that provide lexical evidence for the frames and the corresponding lexical units.
- Two small corpora of *full-text annotation*. These corpora were recently added to the FrameNet package to facilitate statistical analysis of frame-semantic structures in real text.

While the primary purpose of FrameNet is lexical-semantic research, it is widely conjectured that when it matures, it can play an important role in NLP applications, most obviously when there is a need of template-filling.

3 Automatic Frame-Semantic Analysis

Several research prototypes have been implemented that automatically construct frame-semantic structures from text. They have almost exclusively been statistical systems.

The task of frame-semantic analysis is usually divided into three main subtasks: detection and disambiguation of target words, detection of semantic arguments, and finally classification of arguments (see Figure 1).

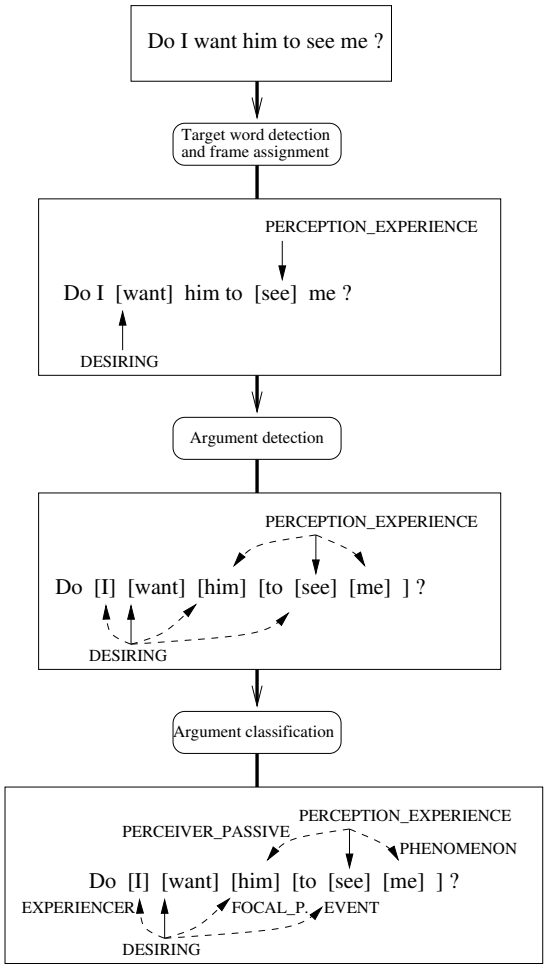

Figure 1: The stages in the frame-semantic structure extraction process.

Although it is generally agreed – from both a linguistic and a machine-learning perspective [22] – that the best performance is obtained when the tasks are solved jointly, it is easiest from an engineering point of view, and computationally less expensive, to treat them as independent tasks that are solved sequentially.

3.1 Implementation of Semantic Analyzers

For both the constituent-based and the dependency-based semantic analyzer, we implemented the three subtasks as statistical classifiers using support vector machines; the first classifier assigns a frame to a given target word, the second decides whether a given node in a constituent or dependency parse tree represents an argument for a given predicate, and the final one assigns a semantic role to a node that has been identified as an argument by the previous step. The second step also makes use of a set of filtering rules to reduce the number of potential arguments [24].

To build statistical models for the classifiers, we collected data from the FrameNet example sentences and the full-text corpora. Although the example sentences are not statistically representative, they contribute significanltly to the lexical coverage as the full-text corpora are small. Since building full statistical models from all the training data is very computationally intensive and takes several days, we used only a fraction of the data to be able to investigate a larger set of parameters.

The following three subsections give an overview of which features are used by the classifiers. Most of the features are shared by the constituent-based and the dependency-based semantic analyzers. Examples of the features are given with respect to the Example 1 and the corresponding constituent and dependency parse trees, shown in Figure 2. The features are also summarized in Table 1. The features used by the constituent-based and the dependency-based classifiers are indicated by C and D, respectively.

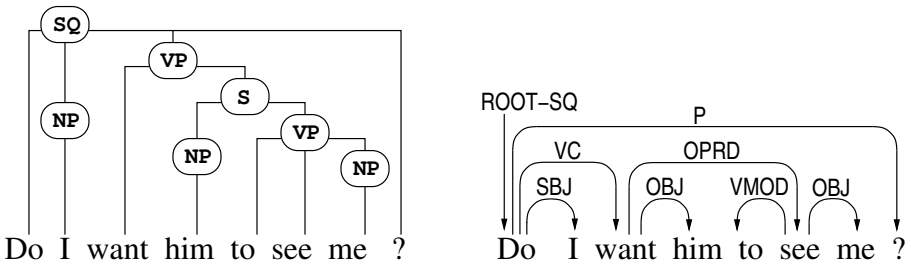

Figure 2: Constituent and dependency trees for the example sentence.

Features	Target disambiguation	Argument identification	Argument classification
Frames	C,D		
TargetLemma	C,D	C,D	C,D
ChildWordSet	C,D		
ParentWord	C,D		
FEs		C,D	C,D
TargetPOS		C,D	C,D
Voice		C,D	C,D
Position		C,D	C,D
ArgWord/POS		C,D	C,D
LeftWord/POS		C,D	C,D
RightWord/POS		C,D	C,D
C-Subcat	C	C	C
C-Path		C	C
PhraseType		C	C
GovCat		C	C
D-Subcat	D	D	D
D-Path		D	D
Function			D
ChildDepSet	D	D	D
ChildWordDepSet	D		

Table 1: Features used by the classifiers.

3.1.1 Common Features

The following features are used by both the constituent-based and the dependency-based semantic analyzers. Head-finding rules [14] were applied when heads of constituents were needed.

Frames. The set of frames listed in FrameNet for a lemma. For instance, for the verb *want*, FrameNet lists Desiring and Possession.

TargetLemma. The lemma of the target word itself, e.g. *want*.

ChildWordSet. The set of dependent head words of the target word. For *see*, this set is { *to*, *me* }.

ParentWord. The word of the parent word of the target. For *see* in the example, this is *want*.

FEs. For a given frame, the set of available frame elements listed in FrameNet. For instance, for *see* in the Perception_experience frame, we have 12 frame elements: Degree, Perceiver_passive, Phenomenon, ...

TargetPOS. Part-of-speech tag for the target word.

VOICE. For verbs, this feature is Active or Passive. For other types of words, it is not defined.

POSITION. Position of the head word of the argument with respect to the target word: Before, After, or On.

HEADWORD and HEADPOS. Word and part-of-speech tag of the argument.

LEFTWORD and LEFTPOS. Word and part-of-speech tag of the leftmost dependent of the argument head.

RIGHTWORD and RIGHTPOS. Word and part-of-speech tag of the rightmost dependent of the argument head.

3.1.2 Features Used by the Constituent-based Analyzer

C-SUBCAT. Subcategorization frame: corresponds to the phrase-structure rule used to expand the phrase around the target. For *want* in the example, this feature is `VP→VB S`.

C-PATH. A string representation of the path through the constituent tree from the target word to the argument constituent. For instance, the path from *want* to *I* is `↑VP-↑SQ-↓NP`.

PHRASETYPE. Phrase type of the argument constituent, e.g. `NP` for *him*.

GOVCAT. Governing category: this feature is either `S` or `VP`, and is found by starting at the argument constituent and moving upwards until either a `VP` or a sentence node (`S`, `SINV`, or `SQ`) is found. For instance, for *him*, this feature is `S`, while for *me*, it is `VP`. This can be thought of as a very primitive way of distinguishing subjects and objects.

3.1.3 Features Used by the Dependency-based Analyzer

D-SUBCAT. Subcategorization frame: the grammatical functions of the dependents concatenated. For *want*, this feature is `OBJ+OPRD`.

D-PATH. A string representation of the path through the dependency tree from the target node to the argument node. Moving upwards through verb chains is not counted in this path string. In the example, the path from *want* to *I* is `↓SBJ`.

FUNCTION. The grammatical function of the argument node. For direct dependents of the target, this feature is identical to the D-PATH.

CHILDDEPSET. The set of grammatical functions of the direct dependents of the target node. For instance, for *want*, this set is { `OBJ`, `OPRD` }.

ChildWordDepSet. The set of word/function pairs of the dependents of the target. For instance, for *want*, this set is { him-OBJ, see-OPRD }.

4 Experiments

The purpose of the experiments is twofold: first, to investigate whether dependency-based semantic analyzers are competitive with constituent-based analyzers; secondly, to measure the influence of the richness of the set of grammatical functions. In addition, we would like to measure the influence of formalisms rather than of parsing performance, so we used two parsers of each type.

To train and evaluate the dependency-based semantic analyzers, we parsed the training and test corpora using two freely available parsers: MaltParser [19] and MSTParser [17]. They were both trained on a dependency treebank that had been automatically converted from the Penn Treebank using the LTH constituent-to-dependency conversion tool [14], achieving a labeled accuracy of 87.4% and 86.9% on section 23 of the WSJ part of the Treebank, respectively. In addition, we trained a model for MaltParser using Penn2Malt [1] which gives a dependency representation with a smaller set of grammatical functions. This model achieves a labeled accuracy of 90.3% on WSJ section 23.

For the constituent-based analyzers, we used the popular Collins' [8] and Charniak's [7] parsers. Although grammatical functions (subject, locative, temporal, ...) are available in the Treebank, they are not available in the output of these parsers. The published labeled precision and recall figures for these parsers are 88.1/87.5 and 89.5/89.6, respectively. Collins' parser comes with three different parsing models; in the experiments, we report the result for the best-performing amongst the three models.

This gives us in total five semantic analyzers that we studied for each experiment: MaltParser and MSTParser with LTH-style dependencies, MaltParser with Penn2Malt dependencies, and Collins' and Charniak's constituent parsers. In the experiments, we studied only target words that were adjectives, adverbs, noun, and verbs, since annotated data are more reliable for these word classes. All the tests were run on the test data from the SemEval-2007 task on Frame-semantic Structure Extraction [2]. The test corpus consists of 120 sentences and contains 970 target words and 1,663 semantic arguments, not counting null-instantiated arguments.

4.1 Target Word Detection and Frame Disambiguation

Table 2 shows the precision, recall, and F1 measures for target word detection and disambiguation. In addition, the last column shows the disambiguation accuracy when the target word is given. When a number appears in boldface,

this denotes that the difference between this figure and the best figure has at least 95% statistical significance according to a McNemar test.

Parser	P	R	F1	Accuracy
Malt/P2M	74.1	68.8	71.4	85.7
Malt/LTH	73.1	67.8	70.3	**84.5**
MST/LTH	73.8	68.4	71.0	85.1
Charniak	74.4	70.1	72.2	85.4
Collins	73.7	68.5	71.1	86.6

Table 2: Target word detection/disambiguation performance.

Interestingly, the performance on this task seems to be negatively affected by the rich set of grammatical functions – the two parsers that use the LTH dependency format score lower than the dependency parser that uses the PENN2MALT format and the two constituent parsers.

Since the dependency-based disambiguation classifiers use more features than their constituent-based counterparts, we believe that this difference may be a case of the "curse of dimensionality" – the large number of features makes the learning curve rise slowly. We intend to carry out a series of feature engineering experiments to investigate this more thoroughly.

4.2 Semantic Argument Detection

In the next experiment, we investigated the performance of argument detection when target words and frames were given. An argument was counted as correctly detected if its bracketing coincided with the bracketing in the gold standard, disregarding punctuation. Table 3 shows the precision, recall, and F1 measures. The table gives results for all targets and for verb targets.

	All targets			Verb targets		
Parser	P	R	F1	P	R	F1
Malt/P2M	62.1	52.0	56.6	64.5	57.5	60.8
Malt/LTH	65.6	53.1	58.7	70.2	58.8	64.0
MST/LTH	57.6	51.2	54.2	58.4	54.9	56.6
Charniak	65.8	51.7	57.9	73.9	61.7	67.3
Collins	63.4	53.0	57.7	73.5	61.1	66.7

Table 3: Semantic argument detection performance.

The most striking discrepancy in the table is the low performance of the semantic analyzer based on MSTPARSER; the differences are much larger than the difference in parsing accuracy on the WSJ. A possible reason for this difference may be that MSTPARSER reportedly performs slightly worse

than MaltParser on short-distance links [16], which possibly would include most arguments.

The other two dependency-based analyzers perform well, especially the model using LTH-style grammatical functions, which outperforms the constituent-based analyzers by 0.8 points for argument detection over all targets.

For verbs, on the other hand, the constituent-based analyzers outperform all the dependency-based ones by a wide margin. Inspection of the output data indicates that a part of the low figures for the dependency-based systems may be explained by ambiguity introduced by coordination and raising. For instance, consider the target word *listen* and the possible argument *He* in the sentences *He watches and I listen* and *He watches and listens*. Parse trees for these sentences are shown in Figure 3. For constituent-based analysis, the system can use the C-Path feature to learn to distinguish the case when *He* is an argument (VP coordination) and the case when it is not (S coordination). The D-Path feature, on the other hand, cannot distinguish the two cases for the dependency-based analyzers.

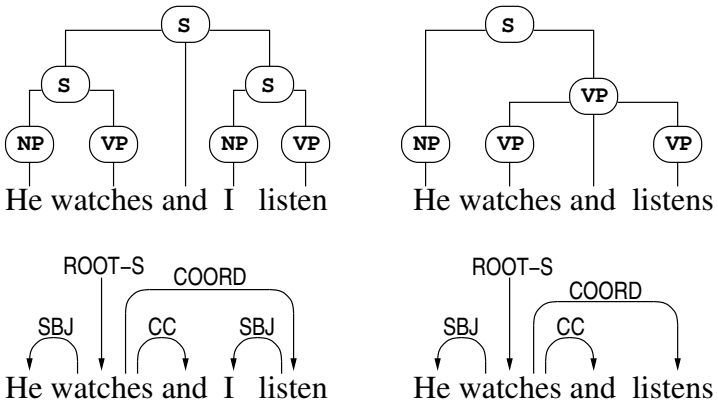

Figure 3: Parse trees of coordination examples.

There are a number of solutions to this problem for the dependency case. First, we could have represented the coordination using the conjunction as head, which would have removed the ambiguity. The second solution would be to distinguish different types of coordination in the dependency trees, such as S-coordination and VP-coordination. Finally, a multi-headed representation could be used, such as in the Danish Dependency Treebank [23]. Whether any of these more complex representations would improve verb argument detection remains to be seen, since they make statistical parsing more difficult. We leave this question to future work.

4.3 Semantic Argument Classification

In the final experiment, we gave the system as input the target words and the corresponding frames, and the argument bracketings. Table 4 shows the argument classification accuracy for all targets and for verb targets. Here, the analyzers based on LTH-style dependencies outperform the constituent-based analyzers. This difference is even more prominent for verbs; we believe that this is because the LTH dependency graphs use 39 different labels for verb arguments and adjuncts, while the PENN2MALT graph only uses five. For non-verb modifiers on the other hand, the LTH and PENN2MALT dependency styles use the same labels.

Parser	All targets	Verb targets
Malt/P2M	69.5	**70.5**
Malt/LTH	69.9	74.2
MST/LTH	70.4	73.6
Charniak	**67.8**	**69.8**
Collins	68.9	72.5

Table 4: Semantic role classification accuracy.

To further assess the influence of grammatical function on the classification accuracy, we performed a feature engineering study. For the dependency-based analyzers, the information about the grammatical function is encoded in the D-PATH and FUNCTION features. Table 5 shows the classification accuracy varies according to the feature set. It is clear that at least one of the D-PATH and FUNCTION features is necessary for accurate classification. Interestingly, it seems that the best performance is achieved when D-PATH is left out. This also reduces the classifier complexity since FUNCTION has a much smaller range than D-PATH.

Feature set			
D-PATH	FUNCTION	All targets	Verb targets
+	+	69.9	74.2
-	+	70.0	75.3
+	-	69.4	74.8
-	-	**68.5**	**72.3**

Table 5: Dependency-based role classification accuracy by feature set.

We performed a similar experiment for constituents (Table 6); here, the information about the grammatical function is encoded in the C-PATH and GOVCAT features. The result suggests that both features are necessary for best accuracy – the C-PATH feature should not be removed, which gives a more complex classifier since the range of this feature is very large.

Feature set			
C-Path	GovCat	All targets	Verb targets
+	+	67.8	69.8
-	+	66.9	70.1
+	-	66.9	69.3
-	-	66.8	**68.5**

Table 6: Constituent-based role classification accuracy by feature set.

5 Perspectives

This study has showed that frame-semantic analysis can be done with roughly the same performance when using dependency syntax as when using constituent syntax. This is a satisfactory result, since for many languages only one type of parser may be available.

The most significant difference seems to be whether the output syntactic structure contains information about grammatical functions or not. In this study, we compared a dependency grammar that had a rich set of functions (39 labels) for verb dependents with one that had a small set (five labels), and showed that the richer set leads to a significant improvement in argument classification accuracy for verbs. For noun and adjective modifiers on the other hand, there was no difference in the set of grammatical functions, and consequently no difference in classification performance.

One direction that was not pursued in this study is to consider constituent trees with labeled edges, as exemplified by the TIGER treebank [5]. Although the Penn Treebank uses grammatical functions, they are not present in the output of the popular constituent parsers. However, they can be can be obtained by applying a post-processing step [4].

References

[1] Penn2Malt. Available on the web at `http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html`.

[2] Collin Baker, Michael Ellsworth, and Katrin Erk. SemEval task 19: Frame semantic structure extraction. In *Proceedings of SemEval*, 2007.

[3] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet Project. In *Proceedings of COLING-ACL*, 1998.

[4] Don Blaheta and Eugene Charniak. Assigning function tags to parsed text. In *Proceedings of NAACL*, 2000.

[5] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER treebank. In *Proceedings of TLT*, 2002.

[6] Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, 2006.

[7] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, 2000.

[8] Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL-EACL*, 1997.

[9] Ronan Collobert and Jason Weston. Fast semantic extraction using a novel neural network architecture. In *Proceedings of ACL*, 2007.

[10] Charles J. Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language*, 280:20–32, 1976.

[11] Daniel Gildea and Julia Hockenmaier. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of EMNLP*, 2003.

[12] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

[13] Daniel Gildea and Martha Palmer. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of ACL*, 2002.

[14] Richard Johansson and Pierre Nugues. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA*, 2007.

[15] 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.

[16] Ryan McDonald and Joakim Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP*, 2007.

[17] Ryan McDonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, 2006.

[18] Igor A. Mel'čuk. *Dependency Syntax: Theory and Practice*. State University Press of New York, Albany, 1988.

[19] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

[20] Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. Semantic role labeling using different syntactic views. In *Proceedings of ACL*, 2005.

[21] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of IJCAI*, 2005.

[22] Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. Joint learning improves semantic role labeling. In *Proceedings of ACL*, 2005.

[23] Matthias Trautner Kromann. The Danish Dependency Treebank and the DTAG treebank tool. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT)*, 2003.

[24] Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*, 2004.