Syntactic features for high precision Word Sense Disambiguation

David Martínez, Eneko Agirre
IxA NLP Group
University of the Basque Country
Donostia, Spain
{jibmaird,eneko}@si.ehu.es

Lluís Màrquez
TALP Research Center
Polytechnical University of Catalonia
Barcelona, Spain
lluism@lsi.upc.es

Abstract

This paper explores the contribution of a broad range of syntactic features to WSD: grammatical relations coded as the presence of adjuncts/arguments in isolation or as subcategorization frames, and instantiated grammatical relations between words. We have tested the performance of syntactic features using two different ML algorithms (Decision Lists and AdaBoost) on the Senseval-2 data. Adding syntactic features to a basic set of traditional features improves performance, especially for AdaBoost. In addition, several methods to build arbitrarily high accuracy WSD systems are also tried, showing that syntactic features allow for a precision of 86% and a coverage of 26% or 95% precision and 8% coverage.

1. Introduction

Supervised learning has become the most successful paradigm for Word Sense Disambiguation (WSD). This kind of algorithms follows a two-step process:

1. Choosing the representation as a set of features for the context of occurrence of the target word senses.
2. Applying a Machine Learning (ML) algorithm to train on the extracted features and tag the target word in the test examples.

Current WSD systems attain high performances for coarse word sense differences (two or three senses) if enough training material is available. In contrast, the performance for finer-grained sense differences (e.g. WordNet senses as used in Senseval 2 (Preiss & Yarowsky, 2001)) is far from application needs. Nevertheless, recent work (Agirre and Martinez, 2001a) shows that it is possible to exploit the precision-coverage trade-off and build a high precision WSD system that tags a limited number of target words with a predefined precision.

This paper explores the contribution of a broad set of syntactically motivated features that ranges from the presence of complements and adjuncts, and the detection of subcategorization frames, up to grammatical relations instantiated with specific words. The performance of the syntactic features is measured in isolation and in combination with a basic set of local and topical features (as defined in the literature), and using two ML algorithms: Decision Lists (Dlist) and AdaBoost (Boost). While Dlist does not attempt to combine the features, i.e. it takes the strongest feature only, Boost tries combinations of features and also uses negative evidence, i.e. the absence of features.

Additionally, the role of syntactic features in a high-precision WSD system based on the precision-coverage trade-off is also investigated.

The paper is structured as follows. Section 2 reviews the features previously used in the literature. Section 3 defines a basic feature set based on the preceding review. Section 4 presents the syntactic features as defined in our work, alongside the parser used. In section 5 the two ML algorithms are presented, as well as the strategies for the precision-coverage trade-off. Section 6 shows the experimental setting and the results. Finally section 7 draws the conclusions and summarizes further work.

2. Previous work.

Yarowsky (1994) defined a basic set of features that has been widely used (with some variations) by other WSD systems. It consisted on words appearing in a window of ±k positions around the target and bigrams and trigrams constructed with the target word. He used words, lemmas, coarse part-of-speech tags and special classes of words, such as "Weekday". These features have been used by other approaches, with variations such as the size of the window, the distinction

between open class/closed class words, or the pre-selection of significative words to look up in the context of the target word.

Ng (1996) uses a basic set of features similar to those defined by Yarowsky, but they also use syntactic information: verb-object and subject-verb relations. The results obtained by the syntactic features are poor, and no analysis of the features or any reason for the low performance is given.

Stetina et al. (1998) achieve good results with syntactic relations as features. They use a measure of semantic distance based on WordNet to find similar features. The features are extracted using a statistical parser (Collins, 1996), and consist of the head and modifiers of each phrase. Unfortunately, they do not provide a comparison with a baseline system that would only use basic features.

The Senseval-2 workshop was held in Toulouse in July 2001 (Preiss & Yarowsky, 2001). Most of the supervised systems used only a basic set of local and topical features to train their ML systems. Regarding syntactic information, in the Japanese tasks, several groups relied on dependency trees to extract features that were used by different models (SVM, Bayes, or vector space models). For the English tasks, the team from the University of Sussex extracted selectional preferences based on subject-verb and verb-object relations. The John Hopkins team applied syntactic features obtained using simple heuristic patterns and regular expressions. Finally, WASP-bench used finite-state techniques to create a grammatical relation database, which was later used in the disambiguation process. The papers in the proceedings do not provide specific evaluation of the syntactic features, and it is difficult to derive whether they were really useful or not.

3. Basic feature set

We have taken a basic feature set widely used in the literature, divided in topical features and local features (Agirre & Martinez, 2001b).

Topical features correspond to open-class lemmas that appear in windows of different sizes around the target word. In this experiment, we used two different window-sizes: 4 lemmas around the target (coded as `win_lem_4w`), and the lemmas in the sentence plus the 2 previous and 2 following sentences (`win_lem_2s`).

Local features include bigrams and trigrams (coded as `big_`, `trig_` respectively) that contain the target word. An index (+1, -1, 0) is used to indicate the position of the target in the bigram or trigram, which can be formed by part of speech, lemmas or word forms (`wf`, `lem`, `pos`). We used TnT (Brants, 2000) for PoS tagging.

For instance, we could extract the following features for the target word `known` from the sample sentence below: word form "`whole`" occurring in a 2 sentence window (`win_wf_2s`), the bigram "`known widely`" where target is the last word (`big_wf_+1`) and the trigram "`RB RB N`" formed by the two PoS before the target word (`trig_pos_+1`).

"There is nothing in the whole range of human experience more widely **known** and universally …"

4. Set of Syntactic Features.

In order to extract syntactic features from the tagged examples, we needed a parser that would meet the following requirements: free for research, able to provide the whole structure with named syntactic relations (in contrast to shallow parsers), positively evaluated on well-established corpora, domain independent, and fast enough.

Three parsers fulfilled all the requirements: *Link Grammar* (Sleator and Temperley, 1993), *Minipar* (Lin, 1993) and (Carroll & Briscoe, 2001). We installed the first two parsers, and performed a set of small experiments (John Carroll helped out running his own parser). Unfortunately, we did not have a comparative evaluation to help choosing the best. We performed a little comparative test, and all parsers looked similar. At this point we chose *Minipar* mainly because it was fast, easy to install and the output could be easily processed. The choice of the parser did not condition the design of the experiments (cf. section 7).

From the output of the parser, we extracted different sets of features. First, we distinguish between direct relations (words linked directly in the parse tree) and indirect relations (words that are two or more dependencies apart in the syntax tree, e.g. heads of prepositional modifiers of a verb). For example, from "`Henry was listed on the petition as the mayor's attorney`" a direct verb-object relation is extracted between `listed`

and Henry and the indirect relation "head of a modifier prepositional phrase" between listed and petition. For each relation we store also its inverse. The relations are coded according to the Minipar codes (cf. Appendix):

```
[Henry    obj_word                    listed]
[listed   objI_word                   Henry]
[petitionmod_Prep_pcomp-n_N_word      listed]
[listed   mod_Prep_pcomp-n_NI_word    petition]
```

For instance, in the last relation above, mod_Prep indicates that listed has some prepositional phrase attached, pcomp-n_N indicates that petition is the head of the prepositional phrase, I indicates that it is an inverse relation, and word that the relation is between words (as opposed to relations between lemmas).

We distinguished two different kinds of syntactic relations: instantiated grammatical relations (IGR) and grammatical relations (GR).

4.1. Instantiated Grammatical Relations

IGRs are coded as [wordsense relation value] triples, where the value can be either the word form or the lemma. Some examples for the target noun "church" are shown below. In the first example, a direct relation is extracted for the "building" sense, and in the second example an indirect relation for the "group of Christians" sense.

```
Example 1: "...Anglican churches have been
           demolished..."
[Church#2  obj_lem                  demolish]

Example 2: "...to whip men into a surrender to a
           particular churh..."
[Church#1  mod_Prep_pcomp-n_N_lem   surrender]
```

4.2. Grammatical relations

This kind of features refers to the grammatical relation themselves. In this case, we collect bigrams [wordsense relation] and also n-grams [wordsense relation1 relation2 relation3 ...]. The relations can refer to any argument, adjunct or modifier. N-grams are similar to verbal subcategorization frames. At present, they have been used only for verbs. Minipar provides simple subcategorization information in the PoS itself, e.g. V_N_N for a verb taking two arguments. We have defined 3 types of n-grams:

- Ngram1: The subcategorization information included in the PoS data given by Minipar, e.g. V_N_N.
- Ngram2: The subcategorization information in ngram1, filtered by the arguments that actually occur in the sentence.
- Ngram3: Which includes all dependencies in the parse tree.

The three types have been explored in order to account for the argument/adjunct distinction, which Minipar does not always assign correctly. In the first case, Minipar's judgment is taken from the PoS. In the second case the PoS and the relations deemed as arguments are combined (adjuncts are hopefully filtered out, but some arguments might be also discarded). In the third, all relations (including adjuncts and arguments) are considered.

In the example below, the ngram1 feature indicates that the verb has two arguments (i.e. it is transitive), which is an error of Minipar probably caused by a gap in the lexicon. The ngram2 feature indicates simply that it has a subject and no object, and the ngram3 feature denotes the presence of the adverbial modifier "still". Ngram2 and ngram3 try to repair possible gaps in Minipar's lexicon.

```
Example: "His mother was nudging him, but he
was still falling"
[Fall#1  ngram1    V_N_N]
[Fall#1  ngram2    subj]
[Fall#1  ngram3    amodstill+subj]
```

5. ML algorithms.

In order to measure the contribution of syntactic relations, we wanted to test them on several ML algorithms. At present we have chosen one algorithm which does not combine features (Decision Lists) and another which does combine features (AdaBoost).

Despite their simplicity, Decision Lists (Dlist for short) as defined in Yarowsky (1994) have been shown to be very effective for WSD (Kilgarriff & Palmer, 2000). Features are weighted with a log-likelihood measure, and arranged in an ordered list according to their weight. In our case the probabilities have been estimated using the maximum likelihood estimate, smoothed adding a small constant (0.1) when probabilities are zero. Decisions taken with negative values were discarded (Agirre & Martinez, 2001b).

AdaBoost (Boost for short) is a general method for obtaining a highly accurate

classification rule by linearly combining many weak classifiers, each of which may be only moderately accurate (Freund, 1997). In these experiments, a generalized version of the Boost algorithm has been used, (Schapire, 1999), which works with very simple domain partitioning weak hypotheses (decision stumps) with confidence rated predictions. This particular boosting algorithm is able to work efficiently in very high dimensional feature spaces, and has been applied, with significant success, to a number of NLP disambiguation tasks, including word sense disambiguation (Escudero et al., 2000). Regarding parametrization, the smoothing parameter has been set to the default value (Schapire, 1999), and Boost has been run for a fixed number of rounds (200) for each word. No optimization of these parameters has been done at a word level. When testing, the sense with the highest prediction is assigned.

5.1. Precision vs. coverage trade-off.

A high-precision WSD system can be obtained at the cost of low coverage, preventing the system to return an answer in the lowest confidence cases. We have tried two methods on Dlists, and one method on Boost.

The first method is based on a **decision-threshold** (Dagan and Itai, 1994): the algorithm rejects decisions taken when the difference of the maximum likelihood among the competing senses is not big enough. For this purpose, a one-tailed confidence interval was created so we could state with confidence 1 - α that the true value of the difference measure was bigger than a given threshold (named θ). As in (Dagan and Itai, 1994), we adjusted the measure to the amount of evidence. Different values of θ were tested, using a 60% confidence interval. The values of θ range from 0 to 4. For more details check (Agirre and Martinez, 2001b).

The second method is based on **feature selection** (Agirre and Martinez, 2001a). Ten-fold cross validation on the training data for each word was used to measure the precision of each feature in isolation. Thus, the ML algorithm would be used only on the features with precision exceeding a given threshold. This method has the advantage of being able to set the desired precision of the final system.

In the case of Boost, there was no straightforward way to apply the first method. The application of the second method did not yield satisfactory results, so we turned to directly use the support value returned for each decision being made. We first applied a threshold directly on this support value, i.e. discarding decisions made with low support values. A second approximation, which is the one reported here, applies a **threshold over the difference in the support** for the winning sense and the second winning sense. Still, further work is needed in order to investigate how Boost could discard less-confident results.

6. Experimental setting and results.

We used the Senseval-2 data (73 nouns, verbs and adjectives), keeping the original training and testing sets. In order to measure the contribution of syntactic features the following experiments were devised (not all ML algorithms were used in all experiments, as specified): contribution of IGR-type and GR-type relations (Dlist), contribution of syntactic features over a combination of local and topical features (Dlist, Boost), and contribution of syntactic features in a high precision system (Dlist, Boost).

Performance is measured as precision and coverage (following the definitions given in Senseval-2). We also consider F1[1] to compare the overall performance as it gives the harmonic average between precision and recall (where recall is in this case precision times the coverage). F1 can be used to select the best precision/coverage combination (cf. section 6.3).

6.1. Results for different sets of syntactic features (Dlist).

Table 1 shows the precision, coverage and F1 figures for each of the grammatical feature sets as used by the decision list algorithm. Instantiated Grammatical Relations provide very good precision, but low coverage. The only exceptions are verbs, which get very similar precision for both kinds of syntactic relations. Grammatical Relations provide lower precision but higher coverage. A combination of both attains best F1, and is the feature set used in subsequent experiments.

[1] F1=2*precision*recall/(precision+recall). In this case we use recall=precision*coverage.

PoS	IGR			GR			All-syntax		
	Prec.	Cov.	F1	Prec.	Cov.	F1	Prec.	Cov.	F1
A	**81,6**	21,8	29,2	70,1	65,4	55,4	70,7	**68,9**	**57,7**
N	**74,6**	36,0	38,5	65,4	57,6	47,8	67,6	**62,5**	**52,0**
V	**68,6**	32,2	33,4	67,3	41,2	39,2	66,3	**52,7**	**45,4**
Ov.	**72,9**	31,9	35,2	67,1	52,1	46,0	67,7	**59,5**	**50,4**

Table 1: precision and coverage for different sets of syntactic features (percentage).

PoS	MFS	Syntax		Local		Local+Topical (Basic)		Basic + Syntax	
		Dlist	Boost	Dlist	Boost	Dlist	Boost	Dlist	Boost
A	59,0	57,7	**62,6**	66,3	**67,5**	65,3	**66,2**	65,4	**67,7**
N	57,1	52,0	**60,0**	63,6	**65,3**	63,2	**67,9**	63,3	**69,3+**
V	40,3	45,4	**48,5**	**51,6**	50,1	51,0	**51,6**	51,2+	**53,9+**
Ov.	48,2	50,4	**55,2**	**59,4**	59,3	58,5	**60,7**	58,7	**62,5+**

Table 2: F1 results (perc.) for different feature sets. "+" indicates statistical significance over Basic.

6.2. Results for different combinations of features (Dlist, Boost)

Both ML algorithms were used on syntactic features, local features, a combination of local+topical features (also called basic), and a combination of all features (basic+syntax) in turn. Table 2 shows the F1 figures for each algorithm, feature set and PoS.

All in all, Boost is able to outperform Dlist in all cases, except for local features. Syntactic features get worse results than local features.

Regarding the contribution of syntactic features to the basic set, the last two columns in Table 2 show a "+" whenever the difference in the precision over the basic feature set is significant (McNemar's test). Dlist is able to scarcely profit from the additional syntactic features (only significant for verbs). Boost attains significant improvement, showing that basic and syntactic features are complementary.

The difference between the two ML algorithms could be explained by the fact that Dlist is a conservative algorithm in the sense that it only uses the positive information given by the first feature that holds in the test example (abstaining if none of them are applicable). By using a combination of the predictions of several single-feature classifers (using both positive and negative evidence) Boost is able to assign positive predictions to more test examples than Dlist. Since the feature space is more widely covered and given that the classifiers are quite accurate, Boost achieves better recall levels and it is a significantly better algorithm for approaching a 100% coverage WSD system.

6.3. Precision vs. coverage: high precision systems (Dlist, Boost)

Figure 1 shows the results for the three methods to exploit the precision/coverage trade-off in order to obtain a high-precision system. For each method two sets of features have been used: the basic set alone and the combination of both basic and syntactic features.

The figure reveals an interesting behavior for different coverage ranges. In the high coverage range, Boost on basic+syntactic features attains the best performance. In the medium coverage area, the feature selection method for Dlist obtains the best results, also for basic+syntactic features. Finally, in the low coverage and high precision area the decision-threshold method for Dlist is able to reach precisions in the high 90's, with no profit from syntactic features.

The two methods to raise precision for Dlists are very effective. The decision-threshold method obtains constant increase in performance up to 93% precision with 7% coverage. The feature selection method attains 86% precision with 26% coverage using syntactic features, but there is no further improvement.

In this case Dlist is able to obtain extremely good accuracy rates (at the cost of low coverage) restricting to the use of the most predictive features. On the contrary, we have had problems in adjusting the AdaBoost algorithm for obtaining high precision predictions.

The figure also shows, for coverage over 20%, that the syntactic features consistently allow for better results, confirming that syntactic features improve the results of the basic set.

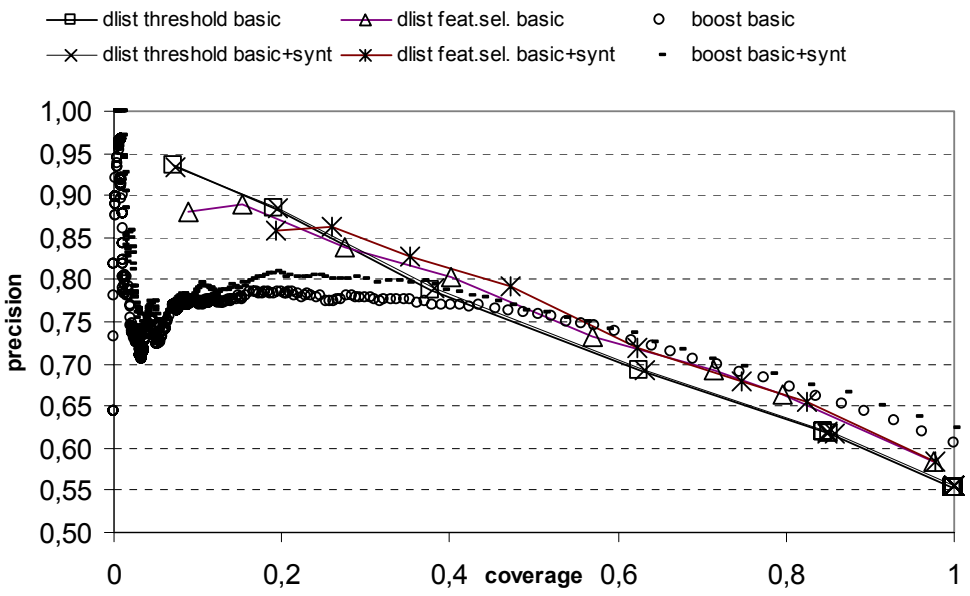

Figure 1: prec./cov. curve for three high precision methods on basic and basic+syntactic features.

7. Conclusions and further work.

This paper shows that syntactic features effectively contribute to WSD precision. We have extracted syntactic relations using the Minipar parser, but the results should be also applicable to other parsers with similar performance. Two kinds of syntactic features are defined: Instantiated Grammatical Relations (IGR) between words, and Grammatical Relations (GR) coded as the presence of adjuncts / arguments in isolation or as subcategorization frames.

The experimental results were tried on the Senseval-2 data, comparing two different ML algorithms (Dlist and Boost) trained both on a basic set of widely used features alone, and on a combination of basic and syntactic features. The main conclusions are the following:

- IGR get better precision than GR, but the best precision/coverage combination (measured with F1) is attained by the combination of both.
- Boost is able to profit from the addition of syntactic features, obtaining better results than Dlist. This proves that syntactic features contain information that is not present in other traditional features.
- Overall the improvement is around two points for Boost, with highest increase for verbs.

Several methods to exploit the precision-coverage trade-off where also tried:

- The results show that syntactic features consistently improve the results on all data points except in the very low coverage range, confirming the contribution of syntax.
- The results also show that Dlist are suited to build a system with high precision: either a precision of 86% and a coverage of 26%, or 95% precision and 8% coverage.

Regarding **future work**, a thorough analysis of the quality of each of the syntactic relations extracted should be performed. In addition, a word-by-word analysis would be interesting, as some words might profit from specific syntactic features, while others might not. A preliminary analysis has been performed in (Agirre & Martinez, 2001b).

Other parsers rather than Minipar could be used. In particular, we found out that Minipar always returns unambiguous trees, often making erroneous attachment decisions. A parser returning ambiguous output could be more desirable. The results of this paper do not depend on the parser used, only on the quality of the output, which should be at least as good as Minipar.

Concerning the performance of the algorithm as compared to other Senseval 2 systems, it is not the best. Getting the best results was not the objective of this paper, but to show that syntactic features are worth including. We plan to improve the pre-processing of our systems, the detection of multiword lexical entries, etc. which could improve greatly the results. In addition there can be a number of factors that could

diminish or disguise the improvement in the results: hand-tagging errors, word senses missing from training or testing data, biased sense distributions, errors in syntactic relations, etc. Factor out this "noise" could show the real extent of the contribution of syntactic features.

On the other hand, we are using a high number of features. It is well known that many ML algorithms have problems to scale to high dimensional feature spaces, especially when the number of training examples is relatively low (as it is the case for Senseval-2 word senses). Researching on more careful feature selection (which is dependent of the ML algorithm) could also improve the contribution of syntactic features, and WSD results in general. In addition, alternative methods to produce a high precision method based on Boost need to be explored.

Finally, the results on high precision WSD open the avenue for acquiring further examples in a bootstrapping framework.

Acknowledgements

This research has been partially funded by McyT (Hermes project TIC-2000-0335-C03-03). David Martinez was funded by the Basque Government, grant AE-BFI:01.245).

References

Agirre, E. and D. Martinez. 2001a. *Decision Lists for English and Basque*. Proceedings of the SENSEVAL-2 Workshop. In conjunction with ACL'2001/EACL'2001. Toulouse, France.

Agirre, E. and D. Martinez. 2001b. *Analysis of supervised word sense disambiguation systems*. Int. report LSI 11-2001, available from the authors.

Brants, T. 2000. *TnT - A Statistical Part-of-Speech Tagger*. In Proc. of the Sixth Applied Natural Language Processing Conference, Seattle, WA.

Carroll, J. and E. Briscoe (2001) `High precision extraction of grammatical relations'. In Proceedings of the 7th ACL/SIGPARSE International Workshop on Parsing Technologies, Beijing, China. 78-89.

Collins M. 1996. *A new statistical parser based on bigram lexical dependencies.* In Proceedings of the 34th Annual Meeting of the ACL, pages 184-191.

Dagan I., and A. Itai. 1994. *Word Sense Disambiguation Using a Second Language Monolingual Corpus.* Computational Linguistics 20:4, pp. 563--596.

Freund Y. and R. E. Schapire. 1997. *A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting*. Journal of Computer and System Sciences, 55(1):119--139.

Escudero G., L. Màrquez, G. Rigau. 2000. *Boosting Applied to Word Sense Disambiguation*. Proceedings of the 12th European Conference on Machine Learning, ECML 2000. Barcelona, Spain.

Kilgarriff, A. and M. Palmer. (eds). 2000. *Special issue on SENSEVAL*. Computer and the Humanities, 34 (1-2).

Lin, D. 1993. *Principle Based parsing without Overgeneration*. In 31st Annual Meeting of the Association for Computational Linguistics. Columbus, Ohio. pp 112-120.

Ng, H. T. and H. B. Lee. 1996. *Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-based Approach.* Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics.

Preiss, J. and D. Yarowsky. 2001. Proc. of the Second Intl. Workshop on Evaluating Word Sense Disambiguation Systems (Senseval 2). In conj. with ACL'2001/EACL'2001. Toulouse, France.

Schapire, R. E. and Y. Singer. 1999. *Improved Boosting Algorithms Using Confidence-rated Predictions*. Machine Learning, 37(3):297--336.

Sleator, D. and D. Temperley. 1993. *Parsing English with a Link Grammar.* Third International Workshop on Parsing Technologies.

Stetina J., S. Kurohashi, M. Nagao. 1998. *General Word Sense Disambiguation Method Based on a Full Sentential Context.* In Usage of WordNet in Natural Language Processing , Proceedings of COLING-ACL Workshop. Montreal (Canada).

Yarowsky, D. 1994. *Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French.* Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, pp. 88--95.

Appendix: main Minipar relations.

Relation	Direct	Indirect	Description
by-subj	X		Subj. with passives
C		X	clausal complement
Cn		X	nominalized clause
comp1	X		complement (PP, inf/fin clause) of noun
Desc	X		description
Fc	X		finite complement
I		X	see c and fc, dep. between clause and main verb
Mod	X		Modifier
Obj	X		Object
pcomp-c	X		clause of pp
Pcomp-n	X		nominal head of pp
Pnmod	X		postnominal modifier.
Pred	X		predicative (can be A or N)
Sc	X		sentential complement
Subj	X		subject
Vrel	X		passive verb modifier of nouns

For each relation the acronym, whether it is used as a direct relation or to construct indirect relations, and a short description are provided.