

Resolving Speculation: MaxEnt Cue Classification and Dependency-Based Scope Rules*

Erik Velldal[♣] and Lilja Øvrelid^{♣♠} and Stephan Oepen[♣]

[♣] University of Oslo, Department of Informatics (Norway)

[♠] Universität Potsdam, Institut für Linguistik (Germany)

erikve@ifi.uio.no and ovrelid@uni-potsdam.de and oe@ifi.uio.no

Abstract

This paper describes a hybrid, two-level approach for resolving hedge cues, the problem of the CoNLL 2010 shared task. First, a maximum entropy classifier is applied to identify cue words, using both syntactic- and surface-oriented features. Second, a set of manually crafted rules, operating on dependency representations and the output of the classifier, is applied to resolve the scope of the hedge cues within the sentence.

1 Introduction

The CoNLL 2010 shared task¹ comprises two sub-tasks. Task 1 is described as *learning to detect sentences containing uncertainty*, while the object of Task 2 is *learning to resolve the in-sentence scope of hedge cues* (Farkas et al., 2010). Paralleling this two-fold task definition, the architecture of our system naturally decomposes into two main steps. First, a maximum entropy (MaxEnt) classifier is applied to automatically detect cue words. For Task 1, a given sentence is labeled as *uncertain* if it contains a word classified as a cue. For Task 2, we then go on to determine the scope of the identified cues using a set of manually crafted rules operating on dependency representations.

For both Task 1 and Task 2, our system participates in the stricter category of ‘closed’ or ‘in-domain’ systems. This means that we do not use any additional uncertainty-annotated material beyond the supplied training data, consisting of 14541 sentences from biomedical abstracts and articles (see Table 2). In the official ranking of re-

sults, and considering systems in all categories together (closed/open/cross-domain), our system is ranked 4 out of 24 for Task 1 and 3 out of 15 for Task 2, resulting in highest average rank (and F_1) overall. We detail the implementation of the cue classifier and the syntactic rules in Sections 3 and 4, respectively. Results for the held-out testing are provided in Section 5. First, however, the next section describes the various resources that we used for pre-processing the CoNLL data sets, to prepare the input to our hedge analysis systems.

2 Architecture and Set-Up

2.1 Preprocessing

To ease integration of annotations across system components, we converted the XML training data to plain-text files, with stand-off annotation linked to the raw data by virtue of character start and end positions (dubbed *characterization* in the following). Thus, hedge cues, scope boundaries, tokenization, Part-of-Speech (PoS) assignments, etc. are all represented in a uniform fashion: as potentially overlapping annotations on sub-strings of the raw input.

The GENIA tagger (Tsuruoka et al., 2005) takes an important role in our pre-processing set-up. However, maybe somewhat surprisingly, we found that its tokenization rules are not always optimally adapted for the BioScope corpus. GENIA unconditionally introduces token boundaries for some punctuation marks that can also occur token-internally. For example, it wrongly splits tokens like ‘3, 926.50’, ‘methlycobamide:CoM’, or ‘Ca(2+)’. Conversely, GENIA fails to isolate some kinds of opening single quotes, because the quoting conventions assumed in BioScope differ from those used in the GENIA Corpus; furthermore, it mis-tokenizes L^AT_EX-style n- and m-dashes.

On average, one in five sentences in the CoNLL

We are grateful to our colleagues at the University of Oslo and the University of Potsdam, for many useful discussions, constructive critique, and encouragement. We specifically thank Woodley Packard for careful proof-reading.

¹The CoNLL 2010 shared task website: <http://www.inf.u-szeged.hu/rgai/conll2010st/>.

ID	FORM	LEMMA	POS	FEATS	HEAD	DEPREL	XHEAD	XDEP
1	The	the	DT	—	4	NMOD	4	SPECDDET
2	unknown	unknown	JJ	degree:attributive	4	NMOD	4	ADJUNCT
3	amino	amino	JJ	degree:attributive	4	NMOD	4	ADJUNCT
4	acid	acid	NN	pers:3 case:nom num:sg ntype:common	5	SBJ	3	SUBJ
5	may	may	MD	mood:ind subcat:MODAL tense:pres clauseType:decl passive:-	0	ROOT	0	ROOT
6	be	be	VB	—	5	VC	7	PHI
7	used	use	VBN	subcat:V-SUBJ-OBJ vtype:main passive:+	6	VC	5	XCOMP
8	by	by	IN	—	7	LGS	9	PHI
9	these	these	DT	deixis:proximal	10	NMOD	10	SPECDDET
10	species	specie	NNS	num:pl pers:3 case:obl common:count ntype:common	8	PMOD	7	OBL-AG
11	.	.	.	—	5	P	0	PUNC

Table 1: Enhanced dependency representation of the example sentence *The unknown amino acid may be used by these species* with GENIAPoS-tags (POS), Malt parses (HEAD, DEPREL) and XLE parses (XHEAD, XDEP).

training data exhibited GENIA tokenization problems. Our pre-processing approach therefore deploys a home-grown, cascaded finite-state tokenizer (borrowed and adapted from the open-source English Resource Grammar; Flickinger (2000)), which aims to implement the tokenization decisions made in the Penn Treebank (Marcus et al., 1993)—much like GENIA, in principle—but properly treating corner cases like the ones above. Synchronized via characterization, this tokenization is then enriched with the output of no less than two PoS taggers, as detailed in the next section.

2.2 PoS Tagging and Lemmatization

For PoS tagging and lemmatization, we combine GENIA (with its built-in, occasionally deviant tokenizer) and TnT (Brants, 2000), which operates on pre-tokenized inputs but in its default model is trained on financial news from the Penn Treebank. Our general goal here is to take advantage of the higher PoS accuracy provided by GENIA in the biomedical domain, while using our improved tokenization and producing inputs to the parsing stage (see Section 2.3 below) that as much as possible resemble the conventions used in the original training data for the parser—the Penn Treebank, once again.

To this effect, for the vast majority of tokens we can align the GENIA tokenization with our own, and in these cases we typically use GENIA PoS tags and lemmas (i.e. base forms). For better normalization, we downcase base forms for all parts of speech except proper nouns. However, GENIA does not make a PoS distinction between proper and common nouns, as in the Penn Treebank, and hence we give precedence to TnT outputs for tokens tagged as nominal by both taggers. Finally, for the small number of cases where we cannot establish a one-to-one alignment from an element

in our own tokenization to a GENIA token, we rely on TnT annotation only. In the merging of annotations across components, and also in downstream processing we have found it most convenient to operate predominantly in terms of characterization, i.e. sub-strings of the raw input that need not align perfectly with token boundaries.

2.3 Dependency Parsing with LFG Features

For syntactic parsing we employ a data-driven dependency parser which incorporates the predictions from a large-scale LFG grammar. A technique of *parser stacking* is employed, which enables a data-driven parser to learn from the output of another parser, in addition to gold standard treebank annotations (Nivre and McDonald, 2008). This technique has been shown to provide significant improvements in accuracy for both English and German (Øvrelid et al., 2009), and a similar approach employing an HPSG grammar has been shown to increase domain independence in data-driven dependency parsing (Zhang and Wang, 2009). For our purposes, we decide to use a parser which incorporates analyses from two quite different parsing approaches—data-driven dependency parsing and “deep” parsing with a hand-crafted grammar—providing us with a range of different types of linguistic features which may be used in hedge detection.

We employ the freely available MaltParser (Nivre et al., 2006), which is a language-independent system for data-driven dependency parsing.² It is based on a deterministic parsing strategy in combination with treebank-induced classifiers for predicting parse transitions. It supports a rich feature representation of the parse history in order to guide parsing and may easily be extended to take into account new features of the

²See <http://maltparser.org>.

	Sentences	Hedged Sentences	Cues	Multi-Word Cues	Tokens	Cue Tokens
Abstracts	11871	2101	2659	364	309634	3056
Articles	2670	519	668	84	68579	782
Total	14541	2620	3327	448	378213	3838

Table 2: Some descriptive figures for the shared task training data. Token-level counts are based on the tokenization described in Section 2.1.

parse history.

Parser stacking The procedure to enable the data-driven parser to learn from the grammar-driven parser is quite simple. We parse a treebank with the XLE platform (Crouch et al., 2008) and the English grammar developed within the ParGram project (Butt et al., 2002). We then convert the LFG output to dependency structures, so that we have two parallel versions of the treebank – one gold standard and one with LFG-annotation. We extend the gold standard treebank with additional information from the corresponding LFG analysis and train the data-driven dependency parser on the enhanced data set. See Øvrelid et al. (2010) for details of the conversion and training of the parser.

Table 1 shows the enhanced dependency representation of the English sentence *The unknown amino acid may be used by these species*, taken from the training data. For each token, the parsed data contains information on the surface form, lemma, and PoS tag, as well as on the head and dependency relation in columns 6 and 7. The dependency analysis suggested by XLE is contained in columns 8 and 9, whereas additional XLE information, such as morphosyntactic properties like number and voice, as well as more semantic properties detailing, e.g., subcategorization frames, semantic conceptual categories such as human, time and location, etc., resides in the FEATS column. The parser outputs, which in turn form the basis for our scope resolution rules discussed in Section 4, also take this same form.

The parser employed in this work is trained on the Wall Street Journal sections 2–24 of the Penn Treebank, converted to dependency format (Johansson and Nugues, 2007) and extended with XLE features, as described above. Parsing is performed using the arc-eager mode of MaltParser (Nivre, 2003) and an SVM with a polynomial kernel. When tested using 10-fold cross-validation on this data set, the parser achieves a labeled accuracy

score of 89.8 (Øvrelid et al., 2010).

3 Identifying Hedge Cues

For the task of identifying hedge cues, we developed a binary maximum entropy (MaxEnt) classifier. The identification of cue words is used for (i) classifying sentences as certain/uncertain (Task 1), and (ii) providing input to the syntactic rules that we later apply for resolving the in-sentence scope of the cues (Task 2). We also report evaluation scores for the sub-task of cue detection in isolation.

As annotated in the training data, it is possible for a hedge cue to span multiple tokens, e.g. as in *whether or not*. The majority of the multi-word cues in the training data are very infrequent, however, most occurring only once, and the classifier itself is not sensitive to the notion of multi-word cues. A given word token in the training data is simply considered to be either a cue or a non-cue, depending on whether it falls within the span of a cue annotation. The task of determining whether a cue word forms part of a larger multi-word cue, is performed by a separate post-processing step, further described in Section 3.2.

3.1 Maximum Entropy Classification

In the MaxEnt framework, each training example—in our case a paired word and label $\langle w_i, y_i \rangle$ —is represented as a feature vector $f(w_i, y_i) = f_i \in \mathbb{R}^d$. Each dimension or feature function f_{ij} can encode arbitrary properties of the data. The particular feature functions we are using for the cue identification are described under Section 3.4 below. For model estimation we use the TADM³ software (Malouf, 2002). For feature extraction and model tuning, we build on the experimentation environment developed by Velldal (2008) (in turn extending earlier work by

³Toolkit for Advanced Discriminative Modeling; available from <http://tadm.sourceforge.net/>.

Oepen et al. (2004)). Among other things, its highly optimized feature handling—where the potentially expensive feature extraction step is performed only once and then combined with several levels of feature caching—make it computationally feasible to perform large-scale ‘grid searches’ over different configurations of features and model parameters when using many millions of features.

3.2 Multi-Word Cues

After applying the classifier, a separate post-processing step aims to determine whether tokens identified as cue words belong to a larger multi-word cue. For example, when the classifier has already identified one or more of the tokens in a phrase such as *raises the possibility* to be part of a hedge cue, a heuristic rule (viz. basically lemma-level pattern-matching, targeted at only the most frequently occurring multi-word cues in the training data) makes sure that the tokens are treated as part of one and the same cue.

3.3 Model Development, Data Sets and Evaluation Measures

While the training data made available for the shared task consisted of both abstracts and full articles from the BioScope corpus (Vincze et al., 2008), the test data was pre-announced to consist of biomedical articles only. In order to make the testing situation during development as similar as possible to what could be expected for the held-out testing, we only tested on sentences taken from the articles part of the training data. When developing the classifiers we performed 10-fold training and testing over the articles, while always including all sentences from the abstracts in the training set as well. Table 2 provides some basic descriptive figures summarizing the training data.

As can be seen in Table 3, we will be reporting precision, recall and F-scores for three different levels of evaluation for the cue classifiers: the sentence-level, token-level and cue-level. The sentence-level scores correspond to Task 1 of the shared task, i.e. correctly identifying sentences as being *certain* or *uncertain*. A sentence is labeled *uncertain* if it contains at least one token classified as a hedge cue. The token-level scores indicate how well the classifiers succeed in identifying individual cue words (this score does not take into account the heuristic post-processing rules for finding multi-word cues). Finally, the cue-level

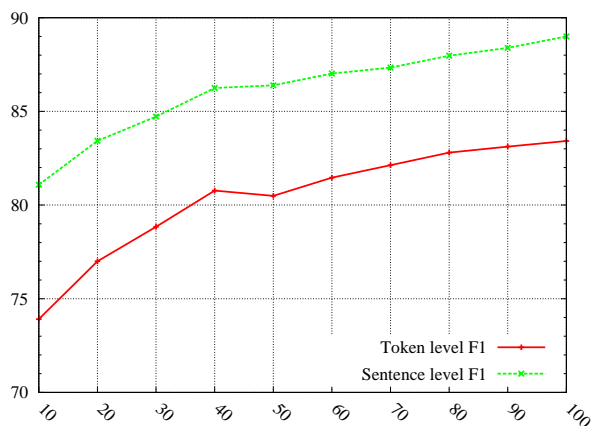


Figure 1: Learning curves showing, for both token- and sentence-level F-scores, the effect of incrementally including a larger percentage of training data into the 10-fold cycles. (As described also for the other development results, while we are training on both the articles and the abstracts, we are testing only on the articles.)

scores are based on the exact-match counts for full hedge cues (possibly spanning multiple tokens). These latter scores are computed using the official shared task scorer script.

3.4 Feature Types

We trained cue classifiers using a wide variety of feature types, both syntactic and surface-oriented. However, to better assess the contribution of the different features, we first trained two baseline models using only features defined for non-normalized surface forms as they occur in the training data. The most basic baseline model (Baseline 1) included only unigram features. The behavior of this classifier is similar to what we would expect from simply compiling a list of cue words from the training data, based on the majority usage of each word as cue or non-cue. Baseline 2 additionally included 2 words to the left and 3 to the right of the focus word (after first performing a search for the optimal spans of n -grams up to 5). As shown in Table 3, this model achieved a sentence-level F1 of 87.14 and a token-level F1 of 81.97. The corresponding scores for Baseline 1 are 79.20 and 69.59.

A general goal in our approach to hedge analysis is to evaluate the contribution of syntactic information, both in cue detection and scope resolution. After applying the parser described in Section 2.3, we extracted a range of classifier features

on the basis of the dependency structures (both as proposed by the stacked MaltParser and converted from XLE) as well as the deep grammar (XLE). Additionally we defined various features on the basis of base forms and PoS information provided by the GENIA pre-processing (see Section 2.2).

For a quick overview, the feature types we experimented with include the following:

GENIA features n -gram features over the base forms and PoS tags from the GENIA information described in Section 2.2.

Dependency features A range of features extracted from dependency structures produced by MaltParser and XLE (see Section 2.3), designed to capture the syntactic properties and environment of a token: **deprel** – dependency relation (Malt and XLE), **deppath** – dependency path to root, **deppattern** – ordered set of co-dependents/siblings, including focus token (Malt), **lextriple/posttriple** – lexicalized and unlexicalized dependency triplet for token (Malt), **coord** – binary feature expressing coordination (XLE), **coordLevel** – phrase-structural level of coordination (XLE).

Lexical parser features Other features constructed on the basis of the parser output: **subcat** – subcategorization frame for verbs (XLE), **advType** – type of adverbial, e.g. sentence, VP (XLE), **adjType** – adjectival function, e.g. attributive vs. predicative (XLE)

When added to Baseline 2 in isolation, most of these features resulted in a boost in classifier performance. For the dependency-based features, the contribution was more pronounced for *lexicalized* versions of the features. This also points to the fact that lexical information seems to be the key for the task of cue identification, where the model using only n -grams over surface forms proved a strong baseline. As more feature types were added to the classifier together, we also saw a clear trend of diminishing returns, in that many of the features seemed to contribute overlapping information. After several rounds of grid-search over different feature configurations, the best-performing classifier (as used for the shared task) used only the following feature types: n -grams over surface forms (including up to 2 tokens to the right), n -grams over base forms (up to 3 tokens left and right), PoS of the target word, ‘subcat’, ‘coord’, and ‘coordLevel’. The ‘subcat’ feature contains

information taken from XLE regarding the sub-categorization requirements of a verb in a specific context, e.g., whether a verb is modal, takes an expletive subject etc., whereas the coordination features signal coordination (‘coord’) and detail the phrase-structural level of coordination (‘coordLevel’), e.g., NP, VP, etc. This defines the feature set used for the model referred to as *final* in Table 3.

Recall that for Baseline 2, the F-score is 87.14 for the sentence-level evaluation and 81.97 for the token-level. For our best and final feature configuration, the corresponding F-scores are 89.00 and 83.42, respectively. At both the sentence-level and the token-level, the differences in classifier performance were found to be statistically significant at $p < 0.005$, using a two-tailed sign-test. After also applying the heuristic rules for detecting multi-word cues, the cue-level F-score for our final model is 84.60, compared to 82.83 for Baseline 2.

3.5 The Effect of Data Size

In order to assess the effect of the size of the training set, we computed learning curves showing how classifier performance changes as more training data is added. Starting with only 10% of the training data included in the 10-fold cycle, Figure 1 shows the effect on both token level and sentence-level F-scores as we incrementally include larger portions of the available training data.

Unsurprisingly, we see that the performance of the classifier is steadily improving up to the point where 100% of the data is included, and by extrapolating from the curves shown in Figure 1 it seems reasonable to assume that this improvement would continue if more data was available. We therefore tried to further increase the size of the training set by also using the hedge-annotated *clinical reports* that form part of the BioScope corpus. This provided us with an additional 855 hedged sentences. However, the classifiers did not seem able to benefit from the additional training examples, and across several feature configurations performance was found to be consistently lower (though not significantly so). The reason is probably that the type of text is quite different—the clinical reports have a high ratio of fragments and also shows other patterns of cue usage, being somewhat more jargon-based. This seems to underpin the findings of previous studies that hedge cue learners appear quite sensitive to text type (Morante and Daele-

Model	Sentence Level			Token Level			Cue Level		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Baseline 1	79.25	79.45	79.20	77.71	63.41	69.59	77.37	71.70	74.43
Baseline 2	86.83	87.54	87.14	86.86	77.69	81.97	85.34	80.21	82.69
Final	91.39	86.78	89.00	91.20	76.95	83.42	90.18	79.47	84.49

Table 3: Averaged 10-fold cross-validation results on the articles in the official shared task training data, always including the abstracts in the training portion. The model listed as *final* includes features such as n -grams over surface forms and base forms (both left and right), PoS, subcategorization frames, and phrase-structural coordination level. The feature types are further described in Section 3.4.

PoS	Description	Source
CC	Coordinations scope over their conjuncts	M
IN	Prepositions scope over their arguments with its descendants	M
JJ _{attr}	Attributive adjectives scope over their nominal head and its descendants	M
JJ _{pred}	Predicative adjectives scope over referential subjects and clausal arguments, if present	M, X
MD	Modals inherit subject-scope from their lexical verb and scope over their descendants	M, X
RB	Adverbs scope over their heads with its descendants	M
VB _{pass}	Passive verbs scope over referential subjects and the verbal descendants	M, X
VB _{rais}	Raising verbs scope over referential subjects and the verbal descendants	M, X
*	For multi-word cues, the head determines scope for all elements	
*	Back off from final punctuation and parentheses	

Table 4: Overview of dependency-based scope rules with information source (MaltParser or XLE), organized by PoS of the cue.

mans, 2009).

4 Resolving Cue Scope

In our approach to scope resolution we rely heavily on syntactic information, taken from the dependency structures proposed by both MaltParser and XLE, as well as various additional features from the XLE parses relating to specific syntactic constructions.

4.1 Scope Rules

We construct a small set of heuristic rules which define the scope for each cue detected in Stage 1. In the construction of these rules, we made use of the information provided by the guidelines for scope annotation in the BioScope corpus (Vincze et al., 2008) as well as manual inspection of the training data in order to arrive at reasonable scope hypotheses for various types of cues.

The rules take as input a parsed sentence which has been tagged with hedge cues and operate over the dependency structures and additional features provided by the parser. Default scope is set to

start at the cue word and span to the end of the sentence (not including final punctuation), and this scope also provides the baseline for the evaluation of our rules. Table 4 provides an overview of the rules employed for scope resolution.

In the case of multi-word cues, such as *indicate that*, and *either ... or*, which share scope, we need to determine the head of the multi-word unit. We then set the scope of the whole unit to the scope of the head token.

As an example, the application of the rules in Table 4 to the sentence with the parsed output in Table 1 correctly determine the scope of the cue *may* as shown in example (1), using a variety of syntactic cues regarding part-of-speech, argumenthood, voice etc. First, the scope of the sentence is set to default scope. Then the MD rule is applied, which checks the properties of the lexical verb *used*, located through a chain of verbal dependents from the modal verb. Since it is passive (passive: +), initial scope is set to include the cue’s subject (SUBJ) argument with all its descendants (*The unknown amino acid*).

Task 1			Task 2			Cue Detection		
Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
85.48	84.94	85.21	56.71	54.02	55.33	81.20	76.31	78.68

Table 6: Evaluation results for the official held-out testing.

Scope	Prec	Rec	F1
Default w/gold cues	45.21	45.21	45.21
Rules w/gold cues	72.31	72.31	72.31
Rules w/classified cues	68.56	61.38	64.77

Table 5: Evaluation of the scope resolution rules on the training articles, using both gold standard cues and predicted cues. For the row labeled *Default*, the scope for each cue is always taken to span rightward to the end of the sentence. In the rows labeled *Rules*, the scopes have been resolved using the dependency-based rules.

- (1) (The unknown amino acid <may> be **used** by these species).

4.2 Rule Evaluation

Table 5 shows the evaluation of the set of scope rules on the articles section of the data set, using gold standard cues.⁴ This gives us an indication of the performance of the rules, isolated from errors in cue detection.

First of all, we may note that the baseline is a strong one: choosing to extend the scope of a cue to the end of the sentence provides an F-score of 45.21. Given gold standard cue information, the set of scope rules improves on the baseline by 27 percentage points on the articles section of the data set, giving us an F-score of 72.31. Comparing to the evaluation using classified cues (the bottom row of Table 5), we find that the use of automatically assigned cues causes a drop in performance of 7.5 percentage points, to a result of 64.77.

5 Held-Out Testing

Table 6 presents the final results as obtained on the held-out test data, which constitute the official results for our system in the CoNLL 2010 shared

⁴This evaluation was carried out using the official scorer script of the CoNLL shared task. When cue information is kept constant, as in our case, the values for false positives and false negatives will be identical, hence the precision and recall values will always be identical as well.

task. The held-out test set comprises biomedical articles with a total of 5003 sentences (790 of them hedged).

For Task 1 we obtain an F-score of 85.21. The corresponding result for the training data, which is reported as ‘Sentence Level’ in Table 3, is 89.00. Although we experience a slight drop in performance (3.8 percentage points), the system seems to generalize quite well to unseen data when it comes to the detection of sentence-level uncertainty.

For Task 2, the result on the held-out data set is an F-score of 55.33, with quite balanced values for precision and recall, 56.7 and 54.0, respectively. If we compare this to the end-to-end evaluation on the training data, provided in the bottom row of Table 5, we find a somewhat larger drop in performance (9.5 percentage points), from an F-score of 64.77 to the held-out 55.3. There are several possible reasons for this drop. First of all, there might be a certain degree of overfitting of our system to the training data. The held-out data may contain hedging constructions that are not covered by our set of scope rules. Moreover, the performance of the scope rules is also influenced by the cue detection, which is reported in the final columns of Table 6. The cue-level performance of our system on the held-out data set is 78.68, whereas the same evaluation on the training data is 84.49. We find that it is the precision, in particular, which suffers in the application to the held-out data set. A possible strategy for future work is to optimize both components of the Task 2 system, the cue detection and the scope rules, on the entire training set, instead of just on the articles.

6 Conclusions — Outlook

We have described a hybrid, two-level approach for resolving hedging in biomedical text, as submitted for the stricter track of ‘closed’ or ‘in-domain’ systems in the CoNLL 2010 shared task. For the task of identifying hedge cues, we train a MaxEnt classifier, which, for the held-out test data, achieves an F-score of 78.68 on the cue-

level and 85.21 on the sentence-level (Task 1). For the task of resolving the in-sentence scope of the identified cues (Task 2), we apply a set of manually crafted rules operating on dependency representations, resulting in an end-to-end F-score of 55.33 (based on exact match of both cues and scopes). In the official shared task ranking of results, and considering systems in all tracks together, our system is ranked 4 out of 24 for Task 1 and 3 out of 15 for Task 2, resulting in the highest average rank overall. For future work we aim to further improve the cue detection, in particular with respect to multi-word cues, and also continue to refine the scope rules. Instead of defining the scopal rules only at the level of dependency structure, one could also have rules operating on constituent structure—perhaps even combining alternative resolution candidates using a statistical ranker.

References

- Thorsten Brants. 2000. TnT. A statistical Part-of-Speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 224–231, Seattle, WA. Association for Computational Linguistics.
- Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar Project. In *Proceedings of COLING Workshop on Grammar Engineering and Evaluation*, pages 1–7.
- Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2008. XLE documentation. Palo Alto Research Center.
- Richard Farkas, Veronika Vincze, Gyorgy Mora, Janos Csirik, and Gyorgy Szarvas. 2010. The CoNLL 2010 Shared Task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the 14th Conference on Natural Language Learning*, Uppsala, Sweden.
- Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1):15–28.
- Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In Joakim Nivre, Heiki-Jaan Kaalep, and Mare Koit, editors, *Proceedings of NODALIDA 2007*, pages 105–112.
- Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 49–55, Taipei, Taiwan.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English. The Penn Treebank. *Computational Linguistics*, 19:313–330.
- Roser Morante and Walter Daelemans. 2009. Learning the scope of hedge cues in biomedical texts. In *Proceedings of the BioNLP 2009 Workshop*, pages 28–36, Boulder, Colorado.
- Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Meeting of the Association for Computational Linguistics*, pages 950–958, Columbus, Ohio, June.
- Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 2216–2219.
- Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Workshop on Parsing Technologies*, pages 149–160.
- Stephan Oepen, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation*, 2(4):575–596.
- Lilja Øvrelid, Jonas Kuhn, and Kathrin Spreyer. 2009. Improving data-driven dependency parsing using large-scale LFG grammars. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics*, pages 37–40, Singapore.
- Lilja Øvrelid, Jonas Kuhn, and Kathrin Spreyer. 2010. Cross-framework parser stacking for data-driven dependency parsing. *TAL 2010 special issue on Machine Learning for NLP*, 50(3).
- Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun’ichi Tsujii. 2005. Developing a robust Part-of-Speech tagger for biomedical text. In *Advances in Informatics*, pages 382–392. Springer, Berlin, Germany.
- Erik Velldal. 2008. *Empirical Realization Ranking*. Ph.D. thesis, University of Oslo, Institute of Informatics, Oslo.
- Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: Annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the BioNLP 2008 Workshop*, Columbus, OH, USA.
- Yi Zhang and Rui Wang. 2009. Cross-domain dependency parsing using a deep linguistic grammar. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics*, Singapore.