UGroningen: Negation detection with Discourse Representation Structures

Valerio Basile and **Johan Bos** and **Kilian Evang** and **Noortje Venhuizen**
{v.basile,johan.bos,k.evang,n.j.venhuizen}@rug.nl
Center for Language and Cognition Groningen (CLCG)
University of Groningen, The Netherlands

Abstract

We use the NLP toolchain that is used to construct the Groningen Meaning Bank to address the task of detecting negation cue and scope, as defined in the shared task "Resolving the Scope and Focus of Negation". This toolchain applies the C&C tools for parsing, using the formalism of Combinatory Categorial Grammar, and applies Boxer to produce semantic representations in the form of Discourse Representation Structures (DRSs). For *negation cue detection*, the DRSs are converted to flat, non-recursive structures, called Discourse Representation Graphs (DRGs). DRGs simplify cue detection by means of edge labels representing relations. *Scope detection* is done by gathering the tokens that occur within the scope of a negated DRS. The result is a system that is fairly reliable for cue detection and scope detection. Furthermore, it provides a fairly robust algorithm for detecting the *negated event or property* within the scope.

1 Introduction

Nothing is more home to semantics than the phenomenon of *negation*. In classical theories of meaning all states of affairs are divided in two truth values, and negation plays a central role to determine which truth value is at stake for a given sentence. Negation lies at the heart of deductive inference, of which consistency checking (searching for contradictions in texts) is a prime example in natural language understanding.

It shouldn't therefore come as a surprise that detecting negation and adequately representing its scope is of utmost importance in computational semantics. In this paper we present and evaluate a system that transforms texts into logical formulas – using the C&C tools and Boxer (Bos, 2008) – in the context of the shared task on recognising negation in English texts (Morante and Blanco, 2012).

We will first sketch the background and the basics of the formalism that we employ in our analysis of negation (Section 2). In Section 3 we explain how we detect negation cues and scope. Finally, in Section 4 we present the results obtained in the shared task, and we discuss them in Section 5.

2 Background

The semantic representations that are used in this shared task on detecting negation in texts are constructed by means of a pipeline of natural language processing components, of which the backbone is provided by the C&C tools and Boxer (Curran et al., 2007). This tool chain is currently in use semi-automatically for constructing a large semantically annotated corpus, the Groningen Meaning Bank (Basile et al., 2012).

The C&C tools are applied for tagging the data with part-of-speech and super tags and for syntactic parsing, using the formalism of Combinatory Categorial Grammar, CCG (Steedman, 2001). The output of the parser, CCG derivations, form the input of Boxer, producing formal semantic representations in the form of Discourse Representation Structures (DRSs), the basic meaning-carrying structures in the framework of Discourse Representation Theory (Kamp and Reyle, 1993). DRT is a widely accepted formal theory of natural language meaning that has been used to study a wide range of linguistic

*First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 301–309,
Montréal, Canada, June 7-8, 2012.

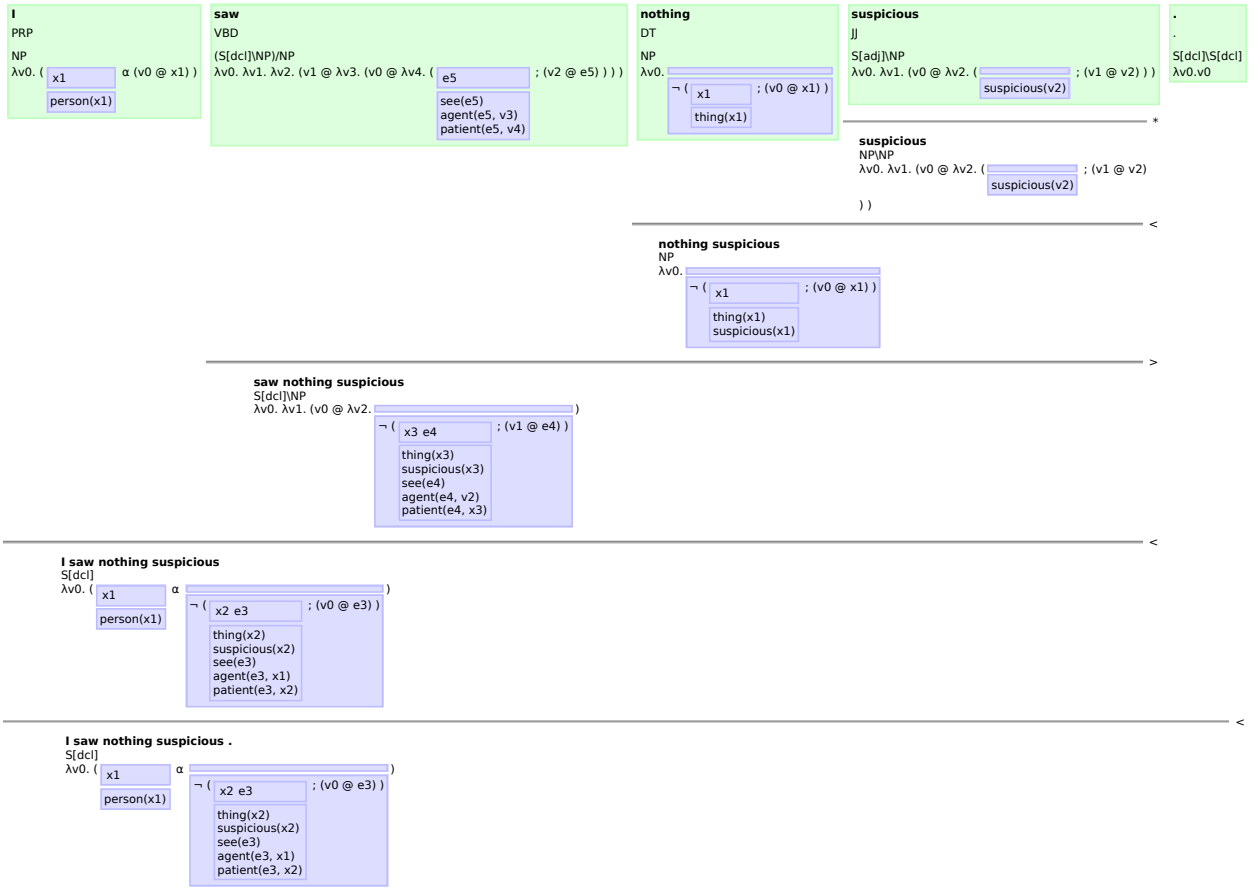

Figure 1: CCG derivation and unresolved semantics for the sentence "I saw nothing suspicious"

phenomena, such as anaphoric pronouns, temporal relations (Kamp and Reyle, 1993), presuppositions (Van der Sandt, 1992), abstract anaphora and rhetorical relations (Asher, 1993; Asher and Lascarides, 2003).

A DRS contains two parts: a set of of discourse referents, and a set of conditions. Negation is represented in a condition by a unary operator in DRT. As an example, Figure 1 shows the derivation for one sentence as produced by the pipeline, illustrating how lexical semantic entries are used to construct a DRS for a whole sentence, guided by the syntactic parse tree. Here, machinery of the λ-calculus is employed to deal with variable renaming when required.

DRSs are recursive structures by nature. They can be produced in several formats (in Prolog or XML) and translated into first-order formulas. The representations can also be generated as a set of tuples, forming together a directed graph equivalent to the original DRS, where discourse referents and symbols are nodes and predicates and relations are viewed as labelled edges. These "flat" Discourse Representation Graphs, DRGs, are often more suitable for certain processing tasks. The tuples also hold additional information, mapping DRS conditions to surface tokens. This mapping is important in tasks where surface realisation plays a role. We also use it in this shared task to get back from complex structures to a flat, token-based annotation of scope.

3 Method

The shared task aims at detecting negation in text — systems are supposed to label tokens that are in the scope of negation, and also identify the token that triggered the negation. The basic idea of our method was to run the existing Boxer system for semantic analysis, then traverse the produced DRSs, and, on encountering an embbeded negated DRS, output the

tokens associated with this negation, as well as the token triggering it.

As this isn't what Boxer is usually asked to do, it required some bookkeeping adjustments. Boxer's anaphora resolution feature was turned off because it is not necessary for the task and would lead to unwanted inclusion of antecedents into negation scopes. Also, its features for representing tense information and rhetorical relations were not used.

The rest of this section pays a closer look at how negation cues are detected and how scope is assigned to tokens. We address the issues of translating a formal representation such as DRS into the format required by the shared task — a representation more oriented at the surface form. We submitted two runs of our system, which both used the C&C tools and Boxer. For the second run, we added some postprocessing steps that tune the result towards a higher performance, especially on scope detection. While these postprocessing steps improve performance, many of them may be specific to the genre and style of the texts used in the shared task.

3.1 Cue detection

Since Boxer has been developed as a system to generate full semantic representations, its lexicon implicitly contains a list of negation cues: those words giving rise to semantic representations of the form $\neg B$, where B is the DRS representing the meaning of the scope of the negation. Key examples here are determiners and noun phrases (*no*, *none*, *no-one*), and verb phrase negation (*not*).

However, negation detection is not the primary function of Boxer, as it is part of the larger aim of providing interpretable semantic representation for English texts, and doing so robustly. So for the current task, after investigating the development data made available by the organisers, Boxer's lexicon was revised at a few points to account for particular negation cues that Boxer originally did not detect. This included the detection of *never* as negation cue, as well as words with a negative prefix or suffix (e.g. *inadequate*, *motionless*). These affix negations were detected using an automatically generated list of negatively affixed nouns, adjectives and adverbs from WordNet (Fellbaum, 1998). The list was created by means of an algorithm that returns all nouns, adjectives and adverbs in WordNet that start with one of *a, an, dis, in, il, im, ir, non, non-, un*, or end with one of *less, lessness, lessly*, and have a direct antonym such that the lemma form equals the stem of the affixed negation (i.e., without the affix).

On the other hand, not everything that introduces a negated DRS in Boxer is a typical negation cue. A case in point is the quantifier *all*, which up until the shared task received a semantics similar to $\lambda P \lambda Q. \neg \exists x(P(x) \wedge \neg Q(x))$ in Boxer's lexicon. As a consequence, Boxer predicted *all* to be a negation cue trigger, in contrast to the shared task gold standard data. Such instances were replaced by logically equivalent representations (in the case of *all*: $\lambda P \lambda Q. \forall x(P(x) \rightarrow Q(x))$).

In order to obtain the tokens that triggered the negated DRS, Boxer's DRG output was used. Occurrences of predicates, relations and connectives in the DRG output carry explicit associations with the tokens in the input whose lexical entries they come from. For basic cue detection, the system annotates as a negation cue those tokens (or affixes) associated with the connective $\neg$ (represented in the DRG as the relation subordinates:neg). Example (1) shows a part of the DRG's tuple format that represents the negation cue "no". *Argument structure* tuples (labeled concept and instance) are also shown, corresponding to a noun in the negation scope, as in "no problem". The first and third columns represent nodes of the DRG graph (both discourse units in this example), the second column represents the label of the edge between the nodes, and the fourth column shows the token associated with the relation (if any).

(1)

...	...	...	...
k1	subordinates:neg	k6	no
k6	concept	c1:problem	
c1:problem	instance	k6:x1	problem
...	...	...	...

In this case, the token "no" is detected as negation cue because it is associated with the relation subordinates:neg.

In the case of affix negation, ideally only the affix should be associated with the negation tuple, and the stem with a corresponding instance tuple. However, since the last column contains tokens, this does not easily fit into the format. We therefore associate the whole affix-negated token with the negation tuple and use separate tuples for affix and stem in order to preserve the information which part of the word

is the cue and which part is in the scope of the negation. The resulting three tuples from a sentence containing the word "injustice" are shown in the following example:

(2)

...	...	...	...
k3	subordinates:neg	k4	injustice
k4	concept	c2:in:71	
k4	concept	c3:justice:1	
...	...	...	...

The target nodes of the two argument structure tuples (labeled concept because "injustice" is a noun) are labeled with the relevant part of the affix-negated word, and a special ID to indicate the presence of a prefix or suffix. This information is used by the script producing the token-based result format. Although multi-word cues, such as *neither...nor* and *on the contrary*, were not correctly predicted as such by Boxer, no effort was made to include them. Due to the token-based detection approach, the cue detection algorithm would have to be severly complicated to include these cases as one negation cue. Because of the relatively small frequency of multi-word cues, we decided not to include special processing steps to account for them.

The second run includes some postprocessing steps implemented on top of the basic output. Since Boxer is not designed to deal with dialogue, interjections were originally ignored as negation cues. Therefore, the postprocessing script added the word "no" as a negation cue (with empty scope) when it occurred as an interjection (tagged "UH"). It also excluded negations with the cue "no" when occurring as part of the expression "no doubt" and not immediately preceded by a verb with the lemma "have" or "be" as in "I have no doubt that...", which *is* to be annotated as a negation. High precision and recall for cue detection on the training data suggested that no further processing steps were worth the effort.

3.2 Scope detection

The tokens in the scope of a negation are determined on the basis of the detected negation cue. It is associated with the negation connective of some negated DRS $\neg B$, so the system annotates as scope all the tokens (and stems in the case of affix negation) directly or indirectly associated with predicates and relations inside B. This includes tokens directly associated with predicates that appear within the negated DRS, as well as those predicates outside of the negated DRS whose discourse referent occurs within the negation scope as the second argument of a thematic role relation.

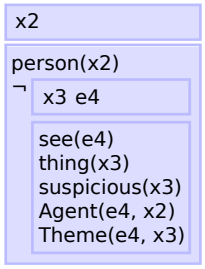

Figure 2: DRS for the sentence "I saw nothing suspicious"

An example is given in Figure 2, where e.g. the tokens *see* and *suspicious* are associated, respectively, with see(e4) and suspicious(x3). Although the predicate person(x2) associated with the pronoun *I* occurs outside of the negated DRS, its referent occurs as an argument within the negated DRS in Agent(e4, x2) and therefore it is taken to be part of the scope of the negation. The desired scope is thus detected, containing the tokens *I*, *saw* and *suspicious*.

Again, in the second run some postprocessing steps were implemented to improve performance. We observed that the scopes in the manually annotated data were usually continuous, except for negation cues within them. However, the scopes produced by the DRS algorithm contained many "gaps" between the tokens of the detected scope, due to an intrinsic feature of the DRS representation. Conventionally, DRSs only explicitly contain content words (i.e. nouns, verbs, adjectives, adverbs), while function words, such as determiners, modals and auxiliary verbs, are represented e.g. as structural properties or temporal features, or not at all, as in the case of the infinitival *to*. Thus, when retrieving the surface representation of the negated scopes from the DRSs, not all structural properties can be directly associated with a surface token and thus not all tokens required for the scope are retrieved. Because in the gold standard annotation these function words were considered part of the negation scope, we designed an ad hoc mechanism to include them, namely filling all the gaps that occur in the negation scope (leaving

out the negation cue). For the same reason, determiners immediately preceding the detected scopes were added in postprocessing. Finally, conjunctions were removed from the beginning of negation scopes, because they were sometimes wrongly recognized by our pipeline as adverbs.

3.3 Negated event/property detection

Although not among our main goals, we also addressed the issue of detecting the "negated event or property" in negation scopes within factual statements. This is done using a heuristic algorithm that uses the detected scope, as well as the syntax tree provided as part of the data.

Since the scope is provided as a set of tokens, the first step is to identify what we call the *scope constituent*, i.e. a constituent in the syntax tree that corresponds to the scope. This is done by going through the tokens in the scope from left to right and determining for each token the largest constituent that starts with this token. The first constituent found in this way the category of whose root is one of SBAR, S and VP is taken to be the scope constituent.

In the second step, the *scope VP* is determined as the first VP encountered when doing a pre-order, left-to-right traversal of the scope constituent. The first verb directly dominated by this VP node determines how the process continues: (i) For non-factual modals (e.g. *may*, *must*, *should*), no event/property is annotated. (ii) For futurity modals (e.g. *would*, *will*, *shall*), the negated event/property is determined recursively by taking the first embedded VP as the new scope VP. (iii) For forms of the verb *be*, the algorithm first looks for the head of an embedded ADJP or NP. If one is found, this is annotated as a negated property. Otherwise, the verb is assumed to be a passive auxiliary and the negated event/property is again determined recursively on the basis of the first embedded VP. (iv) In all other cases, the verb itself is annotated as the negated event.

To limit the undesired detection of negated events/properties outside of factual statements, the algorithm is not applied to any sentence that contains a question mark.

4 Results

Here we discuss our results on the Shared Task as compared to the gold standard annotations provided by (Morante and Daelemans, 2012). The output of our two runs will be discussed with respect to Task 1. The first run includes the results of our system without postprocessing steps and in the second run the system is augmented with the postprocessing steps, as discussed in Section 3.

During the process of evaluating the results of the training data, an issue with the method of evaluation was discovered. In the first version of the evaluation script precision was calculated using the standard formula: $\frac{tp}{tp+fp}$. However, partial matches are excluded from this calculation (they are only counted as false negatives), which means that in the case of *scopes(cue match)*, precision is calculated as the number of exact scope matches (true positives) divided by the number of exact scope matches plus the number of completely wrong instances with no overlap (false positives). As precision is a measure for calculating correctly detected instances among all detected instances, it seems that partial matches should also be taken into account as detected instance. Therefore, we proposed a new evaluation method (B): $\frac{tp}{system}$, where $system$ includes all detected negations of the current system (including partial matches). However, this measure may be too strict as it penalizes a system harder for outputting a partially correct scope than for outputting no scope at all.[1] This choice between two evils seems to indicate that precision is too simple a measure for targets where partial matches are possible. Therefore, in our evaluation of scope detection, we will focus on the *scope tokens* measure where there are no partial matches. For cue and negated event/property detection, we use the stricter, but more meaningful B version. The difference here is almost negligible because these targets typically have just one token.

4.1 Run 1 (without postprocessing)

Table 1 shows the results of the basic system without postprocessing, with the most important results for our system highlighted. As we can see, the basic system performs well on cue detection (F1=

[1] This was pointed out by an anonymous reviewer.

Table 1: Results of the first run (without postprocessing)

Task	gold	system	tp	fp	fn	precision (%)	recall (%)	F1 (%)
Cues:	264	261	219	33	45	86.90	82.95	84.88
Scopes(cue match):	249	261	32	37	217	46.38	12.85	20.12
Scopes(no cue match):	249	261	32	37	217	46.38	12.85	20.12
Scope tokens(no cue match):	1805	1821	1269	552	536	**69.69**	**70.30**	**69.99**
Negated(no cue match):	173	169	89	76	82	53.94	52.05	52.98
Full negation:	264	261	20	33	244	37.74	7.58	12.62
Cues B:	264	261	219	33	45	**83.91**	**82.95**	**83.43**
Scopes B (cue match):	249	261	32	37	217	12.26	12.85	12.55
Scopes B (no cue match):	249	261	32	37	217	12.26	12.85	12.55
Negated B (no cue match):	173	169	89	76	82	52.66	52.05	52.35
Full negation B:	264	261	20	33	244	7.66	7.58	7.62

Table 2: Results of the second run (with postprocessing)

Task	gold	system	tp	fp	fn	precision (%)	recall (%)	F1 (%)
Cues:	264	261	224	28	40	88.89	84.85	86.82
Scopes(cue match):	249	256	102	32	147	76.12	40.96	53.26
Scopes(no cue match):	249	256	102	32	147	76.12	40.96	53.26
Scope tokens(no cue match):	1805	2146	1485	661	320	**69.20**	**82.27**	**75.17**
Negated(no cue match):	173	201	111	85	59	56.63	65.29	60.65
Full negation:	264	261	72	28	192	72.00	27.27	39.56
Cues B:	264	261	224	28	40	**85.82**	**84.85**	**85.33**
Scopes B (cue match):	249	256	102	32	147	39.84	40.96	40.39
Scopes B (no cue match):	249	256	102	32	147	39.84	40.96	40.39
Negated B (no cue match):	173	201	111	85	59	55.22	65.29	59.83
Full negation B:	264	261	72	28	192	27.59	27.27	27.43

83.43%), and reasonably well on the detection of scope tokens (F1$= 69.99\%$).

Note that the results for *Scopes(cue match)* and *Scopes(no cue match)* are the same for our system. Since we make use of token-based cue detection, the only cases of partial cue detection are instances of multi-word cues, which, as discussed above, were not accounted for in our system. In these cases, the part of the cue that is not detected has a large chance of becoming part of the scope of the cue that is detected due to collocation. So, we hypothesize that *Scopes(cue match)* and *Scopes(no cue match)* are the same because in all cases of partial cue detection, the scope incorrectly contains part of the gold-standard cue, which affects both measures negatively.

There is a large discrepancy between the detection of scope tokens and the detection of complete scopes, as the latter is low on both precision (12.26%) and recall (12.85%). The relatively high precision and recall for scope tokens (69.69% and 70.30%, respectively) suggests that there are many cases of partial scope detection, i.e. cases where the scope is either under- or overdetected with respect to the gold standard scope. Since the postprocessing steps for scope detection were developed to reduce exactly this under- and overdetection, we expect that the results for the second run are significantly better. The same holds for negated/event property detection (F1$= 52.98\%$) since it uses the results from scope detection.

4.2 Run 2 (with postprocessing)

Table 2 reports the results of the extended system, which extends the basic system with postprocessing steps for cue detection and especially for scope detection. The postprocessing steps indeed result in higher precision and recall for all tasks, except for *Scope tokens*, which shows a negligible decrease in precision (from 69.69% to 69.20%). This suggests that there are more cases of overdetected scopes than underdetected scopes, because the number of wrongly detected tokens (false positives) increased while the number of undetected scope tokens (false negatives) decreased. This is probably due to the gap-filling mechanism that was implemented as a postprocessing step for scope detection, generaliz-

ing that all scopes should be continuous. We will elaborate more on this point in the discussion in Section 5.

As expected, the detection of complete scopes shows the highest increase in F1 score (from 12.55% to 40.39%). This indicates that the postprocessing steps effectively targeted the weak points of the basic system.

While there are no postprocessing steps for negated event or property detection, the F1 score for this task also increases (from 52.35% to 59.83%), as expected, due to the improvement in scope detection.

5 Discussion

Overall, we can say that both of our systems perform well on cue detection, with a small increase when including the postprocessing steps. This was expected since the postprocessing for cue detection targeted only two specific types of cues, namely, interjections and occurrences of "no doubt". The scope detection benefits consideraby from adding the postprocessing steps, as was their main goal. In the final results of the shared task, run 2 of our system ended second out of five in the open track, while run 1 was ranked last. We will here discuss some points that deserve special attention.

5.1 Affix Negation

As discussed above, affix negations received a special treatment because they were not originally detected as negation cues in Boxer. In the DRS, the token containing the affixed negation cue is associated with two predicates, representing the negative affix and the negated stem. The algorithm secures that only the affix is annotated as the negation cue and that the negated stem is annotated as part of the scope. An example of a sentence containing affix negation is shown in (3) (cardboard 31).[2]

(3) a. [You do yourself an] **in**[justice]. `gold`
b. You do yourself an **in**[justice]. `run1`
c. You do yourself [an] **in**[justice]. `run2`

[2]In the following, boldfaced tokens represent the negation cues, brackets embed the scope and underlining signifies the negated event or property (subscripts added in case of multiple negation cues).

Table 3: Results of negated event/property detection on gold standard cue and scope annotation

Task	prec.(%)	rec.(%)	F1(%)
Negated (no cue match):	64.06	76.88	69.89
Negated B (no cue match):	59.71	76.88	67.22

Note that in neither of the runs the complete scope from the gold standard is detected, although postprocessing increases the recall of scope tokens by adding the determiner "an" to the scope of the negation. However, examples like this are not unambiguous with respect to their negation scope. For example, the sentence in (3) can be interpreted in two ways: "It is *not* the case that you do yourself (something that is) justice" and "It is the case that you do yourself (something that is) *not* justice". While the gold standard annotation assumes the former, wide-scope reading, our system predicts the narrow scope reading for the negation. The narrow scope reading can be motivated by means of Grice's *Maxim of Manner* (Grice, 1975); the choice of an affix negation instead of a verbal negation signals a narrow scope, because in case a wide scope negation is intended, a verbal negation would be more perspicuous. Thus, the difference in the output of our system and the gold standard annotation is in this case caused by a different choice in disambiguating negation scope, rather than by a shortcoming of the detection algorithm.

5.2 Negated event/property detection

Although correct detection of the negated event or property was not our prime concern, the results obtained with our algorithm were quite promising. Among the systems participating in the closed track of task 1, our extended system is ranked third out of seven for negated event/property detection even though the performance on scope detection is lower than all of the other systems in this track. Since negated event/property detection depends on the detected scope, it seems that our heuristic algorithm for detecting the negated event/property is very robust against noisy input. The performance of the detection algorithm on the gold-standard annotation of scopes is shown in Table 3. Although we cannot compare these results to the performance of other systems on the gold standard data, it should be noted

that the results shown here are unmatched by the test results of any other system. It would therefore be worthwile for future work to refine the negated event/property detection algorithm outlined here.

5.3 Postprocessing

The results for the two versions of our system showed that the postprocessing steps implemented in the extended system improved the results considerably, especially for scope detection. Example (4) (cardboard 62) shows the effect of postprocessing on the detection of scopes for negative interjections.

(4) a. "**No**$_1$, [I saw]$_2$ **nothing**$_2$." `gold`
b. "[No], [I saw] **nothing**." `run1`
c. "[**No**$_1$, I saw]$_2$ **nothing**$_2$." `run2`

In Run 1, the system correctly detects the cue "nothing" and the event "saw", although the detected scope is too wide due to an error in the output of the parser we used. In Run 2, postprocessing also correctly recognizes the interjection "no" as a negation cue. Gap filling in this case makes the scope overdetection worse by also adding the comma to the scope. A similar case of the overdetection of scope is shown in (5) (cardboard 85).

(5) a. [The box] is a half-pound box of honeydew tobacco and [does] **not** [help us in any way]. `gold`
b. [The box] is a half-pound box of honeydew tobacco and does **not** [help us in any way]. `run1`
c. [The box is a half-pound box of honeydew tobacco and does] **not** [help us in any way]. `run2`

Note that in Run 1 the first part of the coordinated structure is correctly excluded from the scope of the negation, but the auxiliary "does" is incorrectly not counted as scope. The gap-filling mechanism then adds the intermediary part to the scope of the negation, resulting in an increase in recall for scope tokens detection (since "does" is now part of the scope) but a lower precision because of the overgeneration of the coordinated part.

Nevertheless, the increased precision and recall for scope detection can mainly be ascribed to the gap-filling mechanism implemented in the postprocessing steps for scope detection. As discussed above, the presence of gaps in the original output is due to the non-sequential nature of the DRT representation and the fact that function words are not directly associated with any element in the representations. This suggests that future work on surface realisation from DRSs should focus on translating the structural properties of DRSs into function words.

5.4 Differences between texts

We noted that there was a difference between the performance on text 1 (The Adventure of the Red Circle) and text 2 (The Adventure of the Cardboard Box). The results for text 2 were overall higher than the results for text 1 (except for a 1% decline in recall for *Full negation*). There was a higher scope precision for text 2 and after the postprocessing steps an even larger difference was found for scope detection (15% versus 44% increase in F1 score for *Scopes*). We hypothesize that this difference may be due to a higher number of multiword expressions in text 1 (7 vs. 2) and to the fact that text 1 seems to have more scopes containing gaps. This latter observation is supported by the fact that gap filling results in more overgeneration (more false positives), which is reflected in the ratios of false positives in text 1 (38%) and text 2 (27%). Thus, while the postprocessing steps improve performance, they seem to be genre and style dependent. This motivates further development of the "clean", theoretically motivated version of our system in order to secure domain-independent broad coverage of texts, which is the goal of the Groningen Meaning Bank project.

6 Conclusion

Participating in this shared task on negation detection gave us a couple of interesting insights into our natural language processing pipeline that we are developing in the context of the Groningen Meaning Bank. It also showed that it is not easy to transfer the information about negation from a formal, logical representation of scope to a theory-neutral surface-oriented approach. The results were in line with what we expected beforehand, with the highest loss appearing in the awkward translation from one formalism to another.

References

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Studies in natural language processing. Cambridge University Press.

Nicholas Asher. 1993. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. To appear.

Johan Bos. 2008. Wide-Coverage Semantic Analysis with Boxer. In J. Bos and R. Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 277–286. College Publications.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically Motivated Large-Scale NLP with C&C and Boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic.

Christiane Fellbaum, editor. 1998. *WordNet. An Electronic Lexical Database*. The MIT Press.

H. P. Grice. 1975. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, San Diego, CA.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht.

Roser Morante and Eduardo Blanco. 2012. *SEM 2012 Shared Task: Resolving Scope and Focus of Negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*, Montreal, Canada. To appear.

Roser Morante and Walter Daelemans. 2012. ConanDoyle-neg: Annotation of negation in Conan Doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. To appear.

Mark Steedman. 2001. *The Syntactic Process*. The MIT Press.

Rob Van der Sandt. 1992. Presupposition Projection as Anaphora Resolution. *Journal of Semantics*, 9:333–377.