A pilot study for automatic semantic role labeling in a Dutch corpus

Gerwert Stevens, Paola Monachesi, Antal van den Bosch

Utrecht University, Tilburg University

Abstract

We present an approach to automatic semantic role labeling (SRL) carried out in the context of the D-coi project. Although there has been an increasing interest in automatic SRL in recent years, previous research has focused mainly on English. Adapting earlier research to the Dutch situation poses an interesting challenge especially because there is no semantically annotated Dutch corpus available that can be used as training data. Our automatic SRL approach consists of three steps: bootstrapping from an unannotated corpus with a rule-based tagger developed for this purpose, manual correction and training a machine learning system on the manually corrected data. The input data for our SRL approach consists of Dutch sentences from the D-COI corpus, syntactically annotated by the Dutch dependency parser Alpino.

1 Introduction

The creation of semantically annotated corpora has lagged dramatically behind. As a result, the need for such resources has now become urgent. Several initiatives have been launched at the international level in the last years, however, they have focused almost entirely on English and not much attention has been dedicated to the creation of semantically annotated Dutch corpora.

The Flemish-Dutch STEVIN-program has identified semantic annotation as one of its priorities.[1]

Within the project *Dutch Language Corpus Initiative* (D-Coi), guidelines have been developed for the annotation of a Dutch written corpus. In particular, a 50 million word pilot corpus has been compiled, parts of which have been enriched with (verified) linguistic annotations.[2]

One of the innovative aspects of the D-Coi project is that it has focused not only on the revisions of those protocols which have been already developed within the Spoken Dutch Corpus (CGN) (Oostdijk 2002) for PoS tagging, lemmatization and syntactic annotation but it has also explored the possibility of integrating an additional annotation layer based on semantic information. This annotation layer was not present in the Spoken Dutch Corpus.

One of the goals of the D-Coi project is the development of a protocol for such an annotation layer. In particular, we have dealt with two types of semantic annotation, that is semantic role assignment and temporal and spatial semantics. The reason for this choice lies in the fact that semantic role assignment (i.e. the semantic relationships identified between items in the text such as the agents or patients of particular actions), is one of the most attested and feasible types of semantic

[1] http://taalunieversum.org/taal/technologie/stevin/

[2] http://lands.let.ru.nl/projects/d-coi/

annotation within corpora. On the other hand, temporal and spatial annotation was chosen because there is a clear need for such a layer of annotation in applications like information retrieval or question answering.(Schuurman and Monachesi 2006)

Only a small part of the corpus has be annotated with semantic information, in order to yield information with respect to its feasibility. Hopefully, a more substantial annotation will be carried out in the framework of a follow-up project aiming at the construction of a 500 million word corpus, in which one million words will be annotated with semantic information.

The focus of this paper is on semantic role annotation.[3] We briefly discuss the choices we have made in selecting an appropriate annotation protocol. Furthermore, we present the results of a pilot study for automatic semantic role labeling (SRL) based on the D-coi corpus.

2 Existing projects

During the last few years, corpora enriched with semantic role information have received much attention, since they offer rich data both for empirical investigations in lexical semantics and large-scale lexical acquisition for NLP and Semantic Web applications. Several initiatives are emerging at the international level to develop annotation systems of argument structure, within the D-coi project we have tried to exploit existing results as much as possible and to set the basis for a common standard. We want to profit from earlier experiences and contribute to existing work by making it more complete with our own (language specific) contribution given that most resources have been developed for English.

Within D-coi, the following projects have been evaluated in order to assess whether the approach and the methodology they have developed for the annotation of semantic roles could be adopted for our purposes:

- FrameNet (Johnson, Fillmore, Petruck, Baker, Ellsworth, Ruppenhofer and Wood 2002);
- PropBank (Kingsbury, Palmer and Marcus 2002);

Given the results they have achieved, we have taken their insights and experiences as our starting point.

FrameNet reaches a level of granularity in the specification of the semantic roles which might be desirable for certain applications (i.e. Question Answering). However, it makes automatic annotation of semantic roles rather problematic and might raise problems with respect to uniformity of role labeling even if human annotators are involved. Furthermore, incompleteness constitutes a serious problem, i.e. several frames and relations among frames are missing mainly because FrameNet is still under development. Adopting the FrameNet lexicon for semantic annotation means contributing to its development with the addition of (language specific) and missing frames.

[3] http://www.let.uu.nl/ Paola.Monachesi/personal/DCOI

In our study, we have assumed that the FrameNet classification even though it is based on English could be applicable to Dutch as well. Although Dutch and English are quite similar, there are differences on both sides. For example, in the case of the Spanish FrameNet it turned out that frames may differ in their number of elements across languages (cf. (Subirats and Petruck 2003) and (Subirats and Sato 2004)).

Due to the limitation of available resources, the other alternative was to employ the PropBank approach which has the advantage of providing clear role labels and thus a transparent annotation for both annotators and users. Furthermore, there are promising results with respect to automatic semantic role labeling for English thus the annotation process could be at least semi-automatic. A disadvantage of this approach is that we would have to give up the classification of frames in an ontology, as is the case in FrameNet, which could be very useful for certain applications, especially those related to the Semantic Web. However, in Monachesi and Trapman (2006) suggestions are given on how the two approaches could be reconciled.

A decision was made to adopt a PropBank approach within D-coi mainly because of the prospect of semi-automatic annotation. However, the PropBank annotation guidelines needed to be revised in order to deal with Dutch constructions and with the syntactic annotation layer in D-coi.

Notice that both PropBank and D-coi share the assumption that consistent argument labels should be provided across different realizations of the same verb and that modifiers of the verb should be assigned functional tags. However, they adopt a different approach with respect to the treatment of traces since PropBank creates co-reference chains for empty categories while within D-coi empty categories are almost non existent and in those few cases in which they are attested, a coindexation has been established already at the syntactic level. Furthermore, D-coi assumes dependency structures for the syntactic representation of its sentences while PropBank employs phrase structure trees. In addition, Dutch behaves differently from English with respect to certain constructions and these differences should be spelled out. (Trapman and Monachesi 2006)

3 Automatic SRL

Ever since the pioneering article of Gildea and Jurafsky (2002), there has been an increasing interest in automatic SRL. However, previous research has focused mainly on English. Adapting earlier research to the Dutch situation poses an interesting challenge especially because there is no semantically annotated Dutch corpus available that can be used as training data. Furthermore, no PropBank frame files for Dutch exist.

In PropBank, frame files provide a verb specific description of all possible semantic roles and illustrate these roles by examples. The lack of example sentences makes consistent annotation difficult. Since defining a set of frame files from scratch is very time consuming, we decided to go for an alternative approach, in which we annotated Dutch verbs with the same argument structure as their En-

glish counterparts, thus use English frame files instead of creating Dutch ones. Although this causes some problems, for example, not all Dutch verbs can be translated to a 100% equivalent English counterpart, such problems proved to be relatively rare. In most cases applying the PropBank argument structure to Dutch verbs was straightforward. If translation was not possible, an ad hoc decision was made on how to label the verb.

The second problem, the unavailablity of training data, was partially solved by bootstrapping an unannotated corpus with a rule-based tagger. In short, our automatic SRL approach consists of three steps: bootstrapping from an unannotated corpus with a rule-based tagger, manual correction and finally training a machine learning system on the manually corrected data. The input data for our SRL approach consists of Dutch sentences from the D-COI corpus, syntactically annotated by the Dutch dependency parser Alpino (Bouma, van Noord and Malouf 2000).

Another reason for adopting the PropBank approach was the abstract nature of PropBank argument labeling. Although PropBank roles are not abstract in the sense that different verbs have different role sets, roles are labeled with generic labels: $\text{ARG}_0 \ldots \text{ARG}_n$ and a fixed set of ARGMs. Labels that do not depend on the predicate are an important precondition when building a rule-based system.

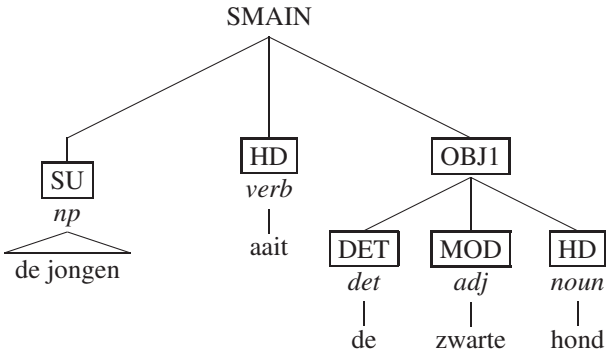
3.1 Dependency structures

Syntactic annotation of the D-Coi corpus is based on the CGN dependency graphs (Moortgat, Schuurman and van der Wouden 2000). A CGN dependency graph is a tree-structured directed acyclic graph in which nodes and edges are labeled with respectively c-labels (category-labels) and d-labels (dependency labels). C-labels of nodes denote phrasal categories, such as NP (noun phrase) and PP, c-labels of leafs denote POS tags. D-Labels describe the grammatical (dependency) relation between the node and its head. Examples of such relations are SU (subject), OBJ (direct object) and MOD (modifier). Figure 1 shows an example of a CGN dependency graph.

There are three main groups of dependency nodes: heads, complements and modifiers. Heads are phrasal heads of the encapsulating syntactic constituent, for example the head noun of a noun phrase. Complements determine the way the thematic structure of the head is interpreted. The most prominent complements are subject and direct object complements. Finally, modifiers mark such notions as time, place and quantity.

Intuitively, dependency structures are a great resource for a rule-based semantic tagger, for they directly encode the argument structure of lexical units, e.g. the relation between constituents. Our goal was to make optimal use of this information in an automatic SRL system. In order to achieve this, we first defined a basic mapping between nodes in a dependency graph and PropBank roles. This mapping forms the basis of our rule-based SRL system.

Figure 1: Example CGN dependency graph



3.2 Mapping dependency structure nodes to PropBank labels

Mapping subject and object complements to PropBank arguments is straightforward: subjects are mapped to ARG0 (proto-typical agent), direct objects to ARG1 (proto-typical patient) and indirect objects to ARG2. An exception is made for ergatives and passives, for which the subject is labeled with ARG1.

Devising a consistent mapping for higher numbered argument is more difficult, since their labeling depends in general on the frame entry of the corresponding predicate. Since we could not use frame information, we used a heuristic method. This heuristic strategy entails that after numbering subject/object complements with the rules stated above, other complements are labeled in a left-to-right order, starting with the first available argument number. For example, if the subject is labeled with ARG0 and there are no object complements, the first available argument number is ARG1.

Examples of complements that can be numbered this way are predictive complements (*Ze schilderde het huis [rood]* 'She painted the house red') and verbal complements (*Ze lijkt [terughoudend te zijn]* 'She seems to be distant').

Finally, a mapping for several types of modifiers was defined. Mapping modifiers consistently is a difficult task due to the fact that their meaning is often ambiguous. For example, the head word *op* ("on") in a prepositional phrase can refer to a location (*Ze loopt op straat* 'She walks on the street') or an indication of manner (*Ze loopt op hoge hakken* 'She walks on high heels'). We refrained ourselves from the disambiguation task, and concentrated on those modifiers that can be mapped consistently. These modifiers are:

- **ArgM-NEC** - Negation markers: lexical units such as *niet* (not), *nooit*

(never) en *geen* (none)

- **ArgM-REC** - Reflexives and reciprocals: lexical units such as *mezelf* (myself) and *zichzelf* (oneself)
- **ArgM-PRD** - Markers of secondary predication: modifiers with the dependency label PREDM
- **ArgM-PNC** - Purpose clauses: modifiers that start with *om te* . These modifiers are marked by Alpino with the c-label OTI.
- **ArgM-LOC** - Locative modifiers: modifiers with the dependency label LD, the LD label is used by Alpino to mark modifiers that indicate a location of direction.

As was demonstrated in this section, thanks to the relational information they contain, it is possible to link PropBank labels to dependency nodes with relatively straightforward mapping rules. This property gives dependency trees an important advantage over phrase structure trees, which are commonly used in SRL systems. The next step in our approach is to implement the mapping rules in a rule-based semantic tagger.

3.3 XARA: a rule based SRL system

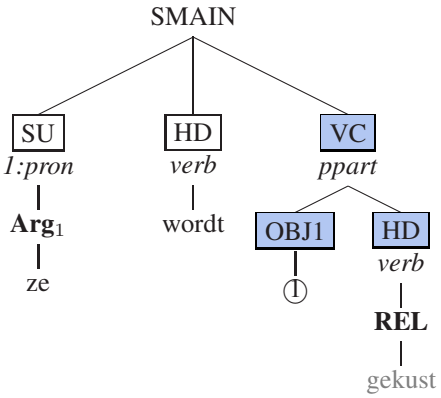
With the help of the mappings discussed above, we developed a rule-based semantic role tagger, which is able to bootstrap an unannotated corpus with semantic roles. We used this rule-based tagger to reduce the manual annotation effort. After all, starting manual annotation from scratch is time consuming and therefore expensive. A possible solution is to start from a (partially) automatically annotated corpus. This reduces the manual annotation task to a manual correction task.

The system we developed for this purpose is called XARA (XML-based Automatic Role-labeler for Alpino-trees) (Stevens 2006). XARA is able to tag a treebank in an XML format with semantic roles. In our experiments we used part of the D-Coi treebank as an input corpus. Dependency trees in this corpus are stored in the Alpino XML format. The structure of Alpino XML documents directly corresponds to the structure of the dependency tree: dependency nodes are represented by NODE elements, attributes of the node elements are the c-label, d-label, pos-tag, etc. The format is designed to support a range of linguistic queries on the dependency trees in XPath directly (Bouma and Kloosterman 2002). XPath (Clark and DeRose 1999) is a powerful query language for the XML format and it is the cornerstone of XARA's rule-based approach.

3.3.1 Rules

A rule in XARA consist of an XPath expression that addresses a node in the dependency tree, and a target label for that node, i.e. a rule is a *(path,label)* pair. For example, a rule that selects direct object nodes and labels them with ARG1 can be formulated as:

Figure 2: Example PropBank annotation on a Dependency tree




```
(//node[@rel='obj1'], 1)
```

XARA supports three types of target labels. In this example, a positive integer is used. Integer labels are used to label nodes with numbered arguments (ARG$_n$). Secondly, for other semantic roles, such as modifiers, string values can be used. Thirdly, the special value -1 can be specified to label the target node with the first available numbered argument; this implements the heuristic labeling strategy described in the previous section.

After their definition, rules can be applied to local dependency domains, i.e. subtrees of a dependency tree. The local dependency domain to which a rule is applied, is called the rule's context. A context is defined by an XPath expression that selects a group of nodes. Contexts for which we defined rules in XARA are verbal domains, that is, local dependency structures with a verb as head. Figure 2 shows an example of such a context: a verbal particle. The nodes that belong to this context are dark colored.

Upon application of a rule, an attribute ("pb") is added to the target node element in the XML file. This attribute contains the PropBank label.

The combination XML + XPath proved to be a very powerful combination for the semantic annotation of our treebank. First of all, because we could work directly with the treebank files and did not need to use an intermediary format. Secondly, because XPath provides a convenient and standardized method to query XML files. This enabled us to use standard Java API's. Finally, because XARA is not restricted to a specific treebank format, but can be used on any XML based treebank other than Alpino with relatively little effort. This property satisfies one of the major design criteria of the system: reusability. The only requirement is that

an XML structure is used that supports XPath queries.

3.4 Classification system

The annotation by XARA of our treebank, was manually corrected by one human annotator. We used these manually corrected sentences as training and test data for a SRL classification system. For this learning system we employed a Memory Based Learning (MBL) approach, implemented in the Tilburg Memory based learner (TiMBL) (Daelemans, Zavrel, van der Sloot and van den Bosch 2004). Memory based learning can be described as reasoning on the basis of similarity of new situations to earlier encountered situations. MBL is often categorized as a ”lazy” approach to learning: instances are directly stored in memory, without any abstraction or restructuring, this is in contrast with greedy approaches such as support vector machines.

During classification, unseen examples are compared to instances in the training data. This comparison is done using a *distance metric* $\Delta(X, Y)$. The class assignment is based on the k-nearest neighbors algorithm: the most common class amongst the k most similar training instances is chosen. In case of a tie among categories, a tie breaking resolution method is used. The goal of classification is to assign class labels to a set of instances automatically. Instances represent the items to be classified by means of a set of features and their target classes.

3.5 Features

TiMBL assigns class labels to training instances on the basis of features. The feature set plays an important role in the performance of a classifier, and choosing features is certainly not a trivial task. In choosing the feature set for our system, we mainly looked at previous research, especially systems that participated in the CoNLL shared tasks (Carreras and Màrquez 2005) for semantic role labeling.

However, none of the systems in the CoNLL shared tasks used features extracted from dependency structures. However, Hacioglu (2004) used dependency tree features for classification. Hacioglu’s system was trained and tested on data of the 2004 CoNLL shared task that was converted into dependency trees. Hacioglu classifies his approach as relation-by-relation (R-by-R) semantic role labeling. The basis of this approach is formed by a new treebank of dependency structures called DepBank. To create the DepBank corpus, first constituency trees from the Penn treebank were converted into dependency trees; furthermore, nodes in the dependency trees that cover a semantic argument were augmented with a PropBank label. For sentences with more than one predicate, the same tree was instantiated with different argument labels.

In a sense, Hacioglu’s approach is comparable to our system, since in both approaches features extracted from dependency trees are used. However, there are also some differences:

- Hacioglu does not use a dependency parser to create the dependency trees, instead existing constituent trees are converted to dependency structures.

- In Hacioglu's system, a dependency tree is created for every proposition in the sentence. In our approach, labels from all propositions in a sentence are stored in a single dependency tree.

- Hacioglu only uses features that are typical to dependency trees (such as the head word of the relation). He does not use "traditional" features like phrase type, i.e. features derived from a phrase structure tree.

From features used in previous system and some experimentation with TiMBL, we derived the following feature set. The first group of features describes the predicate (verb):

(1) Predicate stem - The verb stem, provided by Alpino. This feature is analogous to the *verb lemma* feature used in many existing systems.

(2) Predicate voice - A binary feature indicating the voice of the predicate (passive/active). A predicate is considered passive if it is connected to the auxiliary verb *worden* or *zijn* and is a child of a node with c-label PPART (passive particle).

Notice that the predicate's POS tag is not used as a feature in our system, unlike in many existing systems, since all verbs in Alpino trees have the same POS tag: VERB.

The second group of features describes the candidate argument:

(3) Argument c-label - The category label (phrasal tag) of the node, e.g. NP or PP.

(4) Argument d-label - The dependency label of the node, e.g. MOD or SU.

(5) Argument POS-tag - POS tag of the node if the node is a leaf node, null otherwise.

(6) Argument head-word - The head word of the relation if the node is an internal node or the lexical item (word) if it is a leaf.

(7) Argument head-word - The head word of the relation if the node is an internal node or the lexical item (word) if it is a leaf.

(8) Head-word POS tag - The POS tag of the head word.

(9) c-label pattern of argument - The left to right chain of c-labels of the argument and its siblings.

(10) d-label pattern - The left to right chain of d-labels of the argument and its siblings.

(11) c-label & d-label of argument combined - The c-label of the argument concatenated with its d-label.

Information from this feature set that is not available to XARA is: predicate's root, label pattern of candidate argument and argument position. The position feature was added because it is was used in all CoNLL-05 systems (except one) and in the Hacioglu system. The same applies to the the predicate's root (or lemma). The label pattern feature was used in several CoNLL systems and turned out to have a positive effect on the performance of our system.

3.6 Training procedure

The training set consists of predicate/argument pairs encoded in training instances. Each instance contains features of a predicate and its candidate argument. Candidate arguments are nodes (constituents) in the dependency tree. This pair-wise approach is analogous to earlier work by van den Bosch, Canisius, Daelemans, Hendrickx and Sang (2004) and Sang, Canisius, van den Bosch and Bogers (2005).

Using every possible predicate/argument pair would result in a very large instance base that contains many irrelevant instances. This might lead to reduced performance of the classifier and low classification speed. Therefore, several methods were used to reduce the size of the instance base. The first of these methods is to ignore nodes that can never fill an argument role because of their grammatical function, for example verbal particles. The second method is to only consider phrases that are likely to be arguments.

For example, Sang et al. (2005) build instances from verb/phrase pairs from which the phrase parent is an ancestor of the verb. We adopted this approach to dependency trees: only siblings of the verb (predicate) are considered as candidate arguments.

In comparison to experiments in earlier work, we had relatively few training data available: our training set consisted of 2395 sentences. To overcome our data sparsity problem, we trained the classifier using the leave one out (LOO) method (`-t leave_one_out` option in TiMBL). With this option set, every data item in turn is selected once as a test item, and the classifier is trained on all remaining items.

Except for the LOO option, we only used the default TiMBL settings during training, to prevent overfitting because of data sparsity.

4 Results & Evaluation

4.1 Measures

We used three measures for the evaluation of our system: precision, recall and a combined measure: F-Score. Precision is defined as the proportion of predicted arguments that is predicted correctly, recall as the proportion of correctly predicted arguments. The F-Score is the harmonic mean of precision and recall. To measure the performance of the automatic systems, the automatically assigned labels were compared to the labels assigned by a human annotator.

4.2 Results of XARA labeling

Table 1 shows the performance of XARA on our treebank with 2395 sentences.

Table 1: Results of SRL with XARA

Label	**Precision**	**Recall**	**F**$_{\beta=1}$
Overall	65,11%	45,83%	53,80
Arg0	98.97%	94.95%	96.92
Arg1	70.08%	64.83%	67.35
Arg2	47.41%	36.07%	40.97
Arg3	13.89%	6.85%	9.17
Arg4	1.56%	1.35%	1.45
ArgM-LOC	83.49%	13.75%	23.61
ArgM-NEG	72.79%	58.79%	65.05
ArgM-PNC	91.94%	39.31%	55.07
ArgM-PRD	63.64%	26.25%	37.17
ArgM-REC	85.19%	69.70%	76.67

Since XARA's rules cover only a subset of the argument labels, the classifier is able to achieve a much higher recall score than XARA (see table 2). Precision score of the classifier is higher as well, although the difference with XARA is smaller.

Notice the contrast between XARA's performance on lower numbered arguments and ARG4. Manual inspection of the manual labeling reveals that ARG4 arguments often occur in propositions without ARG2 and ARG3 arguments. Since our current heuristic labeling method always chooses the first available argument number, this method will have to be modified in order achieve a better score for ARG4 arguments.

4.3 Results of TIMBL classification

Table 2 shows the performance of the TiMBL classifier on our annotated dependency treebank. This is the same treebank we used to test the XARA role labeling and consists of 2395 sentences. From these sentences, 12113 instances where extracted.

Some general observations can be made regarding these results:

- A sharp drop in precision and recall for higher numbered arguments can be observed: precision for ARG0 is 90.44%, whereas precision for ARG3 is only 21.21%. This can be contributed in part to the low number of training examples with these labels in the corpus. Performance on lower numbered arguments is relatively good however compared to XARA's performance on these arguments.

Table 2: Results of TiMBL classification

Label	**Precision**	**Recall**	**F**$_{\beta=1}$
Overall	70.27%	70.59%	70.43
Arg0	90.44%	86.82%	88.59
Arg1	87.80%	84.63%	86.18
Arg2	63.34%	59.10%	61.15
Arg3	21.21%	19.18%	20.14
Arg4	54.05%	54.05%	54.05
ArgM-ADV	54.98%	51.85%	53.37
ArgM-CAU	47.24%	43.26%	45.16
ArgM-DIR	36.36%	33.33%	34.78
ArgM-DIS	74.27%	70.71%	72.45
ArgM-EXT	29.89%	28.57%	29.21
ArgM-LOC	57.95%	54.53%	56.19
ArgM-MNR	52.07%	47.57%	49.72
ArgM-NEG	68.00%	65.38%	66.67
ArgM-PNC	68.61%	64.83%	66.67
ArgM-PRD	45.45%	40.63%	42.90
ArgM-REC	86.15%	84.85%	85.50
ArgM-TMP	55.95%	53.29%	54.58

- The ARGM label with the highest F-score is ARGM-REC. This is probably due to the fact that the only information needed to assign this label is the head word feature + POS of the head word, which makes classification of ARGM-RECs relatively easy.

- One would expect a better performance on the lower numbered arguments (assuming that the SU and OBJ1 labels are assigned accurately by the Alpino parser). We expect that the performance on these arguments can be improved by adding lexical features (see section 5).

It is difficult to compare our system with existing systems, since our system is the first one to be applied to Dutch texts. Moreover, our data format, data size and evaluation methods (separate test/train/develop sets versus LOO) are different from earlier research. However, to put our results somewhat in perspective, we looked at the performance of state-of-the-art SRL systems for English.

The CoNLL shared tasks provide an excellent source of information on English PropBank SRL systems that use features extracted from binary phrase structure trees. The best performing system that participated in CoNLL 2005 reached an F_1

of 80. There were seven systems with an F_1 performance in the 75-78 range, seven more with performances in the 70-75 range and five with a performance between 65 and 70.

A system that did not participate in the CoNLL task, but still provides interesting material for comparison since it is also based on dependency structures, is the Hacioglu (2004) system. This system scored 85,6% precision, 83,6% recall and 84,6 F_1 on the CoNLL data set, which is even higher than the best results published so far on the PropBank data sets (Pradhan, K., Krugler, Ward, Martin and Jurafsky 2005): 84% precision, 75% recall and 79 F_1. These results support our claim that dependency structures can be very useful in the SRL task.

5 Conclusion & Further work

The results reported here, provide a first insight into the possibilities and problems of semantic role classification in a Dutch corpus based on Alpino dependency structures. Although several improvements can be made, the first results are encouraging.

One possible improvement consists in the addition of semantic features to the feature set used by the classifier. Examples of such features are the subcategorization frame of the predicate and the semantic category (e.g. WordNet synset) of the candidate argument. We expect that such semantic features will improve the performance of the classifier for certain types of verbs and arguments, especially the lower numbered arguments ARG0 and ARG1. For example, a typical type of classification error we encountered was related to verbs that can have a subject position filled by a theme (ARG1) instead of an agent (ARG0), such as *beginnen* (”to begin”):

(1) [Het boek $_{Arg1}$] begint met een korte inleiding.
”The book begins with a short introduction”

Another example of a possible use of lexical semantic information concerns temporal and spatial modifiers (ARGM-TMP and ARGM-LOC respectively). At the moment, the only available lexical information about such modifiers in our feature set, is the head word of the corresponding preposition. In most cases however, the head word alone is not sufficient to disambiguate the preposition’s meaning. For example, the Dutch preposition *over* can either head a phrase indicating a location or a time-span. The semantic category of the neighboring noun phrase might be helpful in such cases to choose the right PropBank label. Thanks to new lexical resources, such as Cornetto (Vossen 2006), and clustering techniques based on dependency structures (van de Cruys 2005), we might be able add lexical semantic information about noun phrases in future research.

Performance of the classifier can also be improved by automatically optimizing the feature set. The optimal set of features for a classifier can be found by employing bi-directional hill climbing (van den Bosch et al. 2004). There is a wrapper script (Paramsearch) available that can be used with TiMBL and sev-

eral other learning systems that implements this approach[4]. In addition, iterative deepening (ID) can be used as a heuristic way of finding the optimal algorithm parameters for TiMBL.

Finally, it would be interesting to see how the classifier would perform on larger collections and new genres of data. The follow-up of the D-Coi project will provide new semantically annotated data to facilitate research in this area.

References

Bouma, G. and Kloosterman, G.(2002), Querying dependency treebanks in xml, *Proceedings of the Third international conference on Language Resources and Evaluation (LREC)*. Gran Canaria.

Bouma, G., van Noord, G. and Malouf, R.(2000), Alpino: wide-coverage computational analysis of dutch.

Carreras, X. and Màrquez, L.(2005), Introduction to the conll-2005 shared task: Semantic role labeling, *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2005)*. Boston, MA, USA.

Clark, J. and DeRose, S.(1999), Xml path language (xpath), *W3C Recommendation 16 November 1999*. URL: http://www.w3.org/TR/xpath.

Daelemans, D., Zavrel, D., van der Sloot, K. and van den Bosch, A.(2004), Timbl: Tilburg memory based learner, version 5.1, reference guide, *ILK Technical Report Series 04-02*, Tilburg University.

Gildea, D. and Jurafsky, D.(2002), Automatic labeling of semantic roles, *Comput. Linguist.* **28**(3), 245–288.

Hacioglu, K.(2004), Semantic role labeling using dependency trees, *Proceedings of COLING-04*. August 2004.

Johnson, C. R., Fillmore, C. J., Petruck, M. R. L., Baker, C. F., Ellsworth, M. J., Ruppenhofer, J. and Wood, E. J.(2002), *FrameNet:Theory and Practice*.

Kingsbury, P., Palmer, M. and Marcus, M.(2002), Adding semantic annotation to the penn treebank, *Proceedings of the Human Language Technology Conference (HLT'02)*.

Monachesi, P. and Trapman, J.(2006), Merging framenet and propbank in a corpus of written dutch, *Proceedings of the workshop Merging and Layering linguistic information (LREC) 2006*. Genoa.

Moortgat, M., Schuurman, I. and van der Wouden, T.(2000), CGN syntactische annotatie, *Internal report Corpus Gesproken Nederlands*.

Oostdijk, N.(2002), The design of the spoken dutch corpus, *in* P. Peters, P. Collins and A. Smith (eds), *New Frontiers of Corpus Research*, pp. 105–112. Amsterdam: Rodopi.

Pradhan, S., K., Krugler, V., Ward, W., Martin, J. H. and Jurafsky, D.(2005), Support vector learning for semantic argument classification, *Machine Learning Journal* **1-3**(60), 11–39.

Sang, E. T. K., Canisius, S., van den Bosch, A. and Bogers, T.(2005), Applying spelling error correction techniques for improving semantic role label-

[4] URL: http://ilk.uvt.nl/software.html#paramsearch

ing, *Proceedings of the Ninth Conference on Natural Language Learning (CoNLL-2005)*. Ann Arbor, MI, USA.

Schuurman, I. and Monachesi, P.(2006), The contours of a semantic annotation scheme for dutch, *Proceedings of Computational Linguistics in the Netherlands 2005*, University of Amsterdam. Amsterdam.

Stevens, G.(2006), *Automatic semantic role labeling in a dutch corpus*, Master's thesis, Universiteit Utrecht.

Subirats, C. and Petruck, M. R. L.(2003), Surprise: Spanish framenet!, *in* E. Hajicova, A. Kotesovcova and J. Mirovsky (eds), *Proceedings of CIL 17*. Prague: Matfyzpress.

Subirats, C. and Sato, H.(2004), Spanish framenet and framesql, *4th International Conference on Language Resources and Evaluation. Workshop on Building Lexical Resources from Semantically Annotated Corpora*. Lisbon (Portugal), May 2004.

Trapman, J. and Monachesi, P.(2006), Manual for the annotation of semantic roles in d-coi, *Technical report*, University of Utecht.

van de Cruys, T.(2005), Semantic clustering in dutch., *Proceedings of CLIN 2005*.

van den Bosch, A., Canisius, S., Daelemans, W., Hendrickx, I. and Sang, E. T. K.(2004), Memory-based semantic role labeling: Optimizing features, algorithm, and output, *in* H. Ng and E. Riloff (eds), *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*. Boston, MA, USA.

Vossen, P.(2006), Cornetto: Een lexicaal-semantische database voor taaltechnologie, *Dixit Special Issue*. Stevin.