Exploiting Grammatical Relations for Protein Relation Extraction and Role Labeling

Timur Fayruzov
Ghent University Association
Krijgslaan 281 (S9)
9000 Gent, Belgium
Timur.Fayruzov@ugent.be

Martine De Cock
Ghent University Association
Krijgslaan 281 (S9)
9000 Gent, Belgium
Martine.DeCock@ugent.be

Chris Cornelis
Ghent University Association
Krijgslaan 281 (S9)
9000 Gent, Belgium
Chris.Cornelis@ugent.be

Veronique Hoste
Ghent University Association
Groot-Brittanniëlaan 45
9000 Gent, Belgium
Veronique.Hoste@hogent.be

ABSTRACT

Automatic protein interaction mining from natural language texts and automatic identification of the agent and target proteins (i.e. role labeling) are challenging problems that attract a lot of attention because of the growing amount of biomedical text resources. We propose a novel approach that relies exclusively on parsing and dependency information. We strategically omit any context information such as keywords or parts-of-speech to maximally abstract from the given corpora and look whether the grammatical relations correspond to the semantic relations in the text and how close this correspondence is. In particular, we construct a feature vector for each sentence only from the grammatical relations and some parsing information. We then use the obtained vector with standard machine learning algorithms in deciding whether a sentence describes a protein interaction and which roles the interaction participants play. Evaluation on benchmark datasets shows that our method is competitive with existing state-of-the-art algorithms for the extraction of protein interactions, and gives promising results for protein role detection.

1. INTRODUCTION

Studying proteins and genes is currently one of the most important directions in biomedical research. Such research not only contributes to the understanding of nature and its structure, but also provides fundamental knowledge about diseases and their origins. Moreover, understanding the functions of genes and proteins and the ways of interaction between them gives to the biologist a means to control certain aspects of live organism development. Due to all these facts, gene and protein interactions research attracts a tremendous amount of attention these days. Consequently, this area accumulates a vast amount of experimental data, which needs to be organized, structured, and shared.

More and more relevant information on protein interactions is becoming available on the web, not only in specialized structured databases such as IntAct[1] and ontological resources such as the Gene Ontology[2], but also in less structured literature databases such as MEDLINE[3]. The former resources are built and maintained manually and thus require a great deal of knowledge and labour intensive maintenance to stay synchronized with the latest research findings in molecular biology, while the latter is always up to date. However, it is not straightforward to query and find specific information in a text database such as MEDLINE, since it is loosely structured and provides very few metadata about its content. Hence it comes as no surprise that machine learning techniques for information extraction (IE) in the biomedical domain have gained a lot of attention over the last years as a means to bridge the gap between the abundance of unstructured textual data on one hand and the structured but limited databases on the other hand.

In this paper, we focus on the problem of finding protein or gene interactions in natural language texts, as in the following sentence: "Most <u>cot</u> genes, and the <u>gerE</u> gene, are transcribed by <u>sigmaK</u> RNA polymerase." The genes are underlined. Firstly, we are interested in the interactions that this sentence bears. It expresses two interactions, one between *cot* and *sigmaK*, and another between *gerE* and *sigmaK*. Secondly, for each interaction we want to detect protein role labels. Interactions can be symmetric assuming that both proteins have equal roles, but can also be asymmetric, e.g. in the sentence above it is clear that *sigmaK* affects both *gerE* and *cot* genes. This means that *sigmaK* is an effector (agent) and *gerE* and *cot* are effectees (targets). Protein role labeling is an important task not only from the biological point of view, but also because it provides a next level of natural language structuring and hence allows to obtain a more precise and detailed knowledge base. The aim

[1] `http://www.ebi.ac.uk/intact/site/index.jsf`
[2] `http://www.geneontology.org/`
[3] `http://www.ncbi.nlm.nih.gov/`

of the relation extraction algorithm presented in this paper is to correctly extract described relations and (optionally) assign proper roles to the interaction participants.

Relations between biological entities can be expressed in many different ways. Hence it is unfeasible to construct a universal set of patterns or rules that will successfully extract interactions from any kind of text. Traditional algorithms for relation learning from texts can perform reasonably well (see e.g. [1, 2, 5, 13]), but they typically rely explicitly or implicitly on specific interaction keywords. This makes them unsuitable for heterogeneous corpora in which protein interactions are described using different lexicons. However, entirely different surface representations for interactions can still exhibit the same syntactic pattern. Moreover, grammatical structures, such as passive voice, can give useful clues for role detection. In this paper we therefore abstract from any lexical data and concentrate on dependency and parsing information, thereby exploring the potential of syntactic data for relation extraction.

The resulting system is a mining tool that facilitates information extraction and knowledge base maintenance by presenting to the user protein interactions with labeled roles identified in scientific texts. The tool aims at supporting biologists in finding relevant information, rather than to exclude them entirely from the data processing flow. After reviewing related approaches in Section 2, we give a description of the proposed method in Section 3. Abstracting from pure lexical data and only relying on syntactic patterns instead, bears the risk of overgeneralizing, in the sense that sentences that do not describe protein interactions might exhibit a syntactic structure similar to those that do, and hence they might get incorrectly identified as protein interactions. To verify the reliability of our approach we therefore evaluate it on two benchmark datasets that are described in Section 4. The experimental results and a comparison with existing algorithms are described in Section 5. Concluding remarks and future work are presented in Section 6.

This paper is a follow-up on the work in [6], where we explored for the first time the use of pure syntactic information for the extraction of gene and protein relations. The novelty of the current paper is that we now also consider protein roles, and extract interactions enriched with this information. This is done successfully using the same technique, which again points out the generality of the proposed approach.

2. RELATED WORK

The extraction of protein relations has attracted a lot of attention during the last years, resulting in a range of different approaches. Throughout this paper we follow the commonly accepted assumption that the relations under consideration are binary, i.e. they have only two participants. Another common simplification is to assume that there are no cross-sentence interactions, i.e., every interaction is described within one sentence. The problem of relation learning can be approached from different points of view, which makes the comparison of results obtained by different researchers problematic. First of all, different approaches are often tailored to work well on different datasets. Secondly, many approaches concentrate only on some particular types of interactions, e.g inhibition [14], or on relations between certain biological entities, e.g. between genes and diseases, genes and proteins, etc. Moreover, some approaches can only be used for protein relation detection while others can be used for role labeling as well.

The recognition of protein interactions is typically treated as a classification problem: the classifier gets as input information about a sentence containing two protein names and decides whether the sentence describes an actual interaction between those proteins or not. When taking role labels into account, the classification task is modified to classify each sentence among three classes — no, agent-target and target-agent interactions. The classifier itself is built manually or, alternatively, it is constructed automatically using an annotated corpus as training data. The different approaches can be distinguished based on the information they feed to the classifier: some methods use only shallow parsing information on the sentence while others exploit full parsing information.

Shallow parsing information includes part-of-speech (POS) tags and lexical information such as lemmas (the base form of words occurring in the sentence) and orthographic features (capitalization, punctuation, numerals etc.). In [2], a support vector machine (SVM) model is used to discover protein interactions. In this approach each sentence is split into three parts — before the first protein, between the two proteins and after the second protein. The kernel function between two sentences is computed based on common sequences of words and POS tags. Since this approach was applied to protein-protein interaction detection, it does not consider asymmetry of the interactions. In another approach that does take the interaction direction into account [5], this kernel function is modified to treat the same parts of the sentence as bags-of-words and called a global context kernel. It is combined with another kernel function called a local context kernel, that represents a window of limited size around the protein names and considers POS, lemmas, and orthographic features as well as the order of words. The resulting kernel function in this case is a linear combination of the global context kernel, and the local context kernel.

A completely different approach is presented in [13], where very high recall and precision rates are obtained by means of hand-crafted rules for sentence splitting and protein relation detection (without considering direction). The rules are based on POS and keyword information, and they were built and evaluated specifically for *Escherichia coli* and yeast domains. It is questionable, however, whether comparable results could be achieved in different biological domains and how much effort would be needed to adapt the approach to a new domain. In another approach reported on in [9], a system was built specifically for the LLL challenge (see Section 4). First, training set patterns are built by means of pairwise sentence alignment using POS tags. Next, a genetic algorithm is applied to build several finite state automata (FSA) that capture the relational information from the training set with respect to interaction direction.

Besides the methods described above, approaches have been proposed that augment shallow parsing information with full parsing information, i.e. syntactic information such as full parse and dependency trees. In [4] for instance, for every sentence containing two protein names a feature vector is built containing terms that occur in the path between the proteins in the dependency tree of the sentence. These feature vectors are used to train an SVM based classifier with a linear kernel. More complex feature vectors are used in [11], where the local contexts of the protein names, the root verbs of the sentence, and the parent of the protein nodes in the dependency tree are taken into account by a BayesNet-based meta-classifier. The resulting classifier is able to distinguish interaction directions. In [8], syntactic information preprocessing, hand-made rules, and a domain vocabulary are used to extract gene interactions. Interaction direction is detected by a special rule, that looks for the passive verb forms. The approach in [19] uses predicate-argument structures (PAS) built from dependency trees. As surface variations may exhibit the same PAS, the approach aims at tackling lexical variance in the data. It is tailored towards the AImed dataset (see Section 4) for which 5 classes of relation expression templates are predefined manually. The classes are automatically populated with examples of PAS patterns and protein interactions are identified by matching them against these patterns. As this dataset does not contain direction information, this approach does not take it into account.

To the best of our knowledge, all existing work either uses only shallow parsing information (including lexical information) or a combination of shallow and full parsing information. Furthermore, approaches of the latter kind typically use dependency trees only as a means to e.g. detect chunks or to extract relevant keywords. The goal of this paper is to investigate what can be achieved using *only* full parsing information. We want to evaluate how far can we go without considering any context information, using only meta-level data. This means that the full parsing information is used not for data preprocessing, but as a direct input itself to the relation extraction classifier. Moreover, we use the same information to mine the protein roles in the extracted interactions. The fact that such an approach is independent of the use of a specific lexicon makes it worthwhile to investigate.

3. SYNTAX-BASED APPROACH

Protein and gene interactions of different types can be expressed in many different ways in English, which makes it impossible to build a corpus that covers all the relation patterns, even for some particular types of interactions. In other words, for a previously unseen text there is a high probability that it contains interactions for which there are no exact matching examples in the training data. However, these interactions may still share some implicit regularities that can be exploited to improve the quality of the interaction extraction process. In our work we appeal to grammatical structures that underlie sentences, i.e., we use syntactic patterns to describe proteins and genes relations. The following example illustrates how at first glance different sentences can share the same syntactic pattern representing an interaction.

Example 1 Consider the following sentences about the interaction between *sigma F* and *sigma E* in one case, and between *GerE* and *cotB* in the other case:

> Sigma F activity regulates the processing of sigma E within the mother cell compartment.

> A low GerE concentration, which was observed during the experiment, activated the transcription of cotB by final sigmaK RNA polymerase, whereas a higher concentration was needed to activate transcription of cotX or cotC.

Although the surface representations are very different, the underlying syntactic pattern, which represents part of a dependency tree, is the same in both cases:

$$protein \overset{nn}{\leftarrow} noun \overset{nsubj}{\leftarrow} verb \overset{dobj}{\rightarrow} noun \overset{prep\_of}{\rightarrow} protein$$

We exploit this deeper similarity between instances by using dependency and parsing information to build abstract representations of interactions. Such representations have less variance than the initial lexical data, hence sensible results can be obtained from smaller training datasets. The approach is fully automated and consists of three stages: after a text preprocessing stage, for every sentence containing two tagged protein names[4], we construct a feature vector summarizing relevant information on the parse tree and the dependency tree. In the third stage, a classifier is trained to recognize whether the sentence describes an actual interaction between the proteins, and to detect which proteins fulfill the role of agents and targets. The novelty of the approach w.r.t. existing work [8, 11, 19] is that we do not use dependency data to detect keywords, but we consider dependencies as features themselves. In Section 5, we show that using only this kind of syntactic information without any lexical data allows to obtain reasonably good results.

3.1 Text preprocessing

The text processing phase includes sentence splitting as well as the detection and the substitution of complex utterances (e.g. chemical formulas or constructions with many parentheses) with artificial strings Furthermore, we expand repeating structures, turning e.g. 'sigA- and sigB-proteins' or 'sigA/sigB-proteins' into 'sigA-proteins and sigB-proteins'. All substitutions are done automatically by means of regular expressions; hence the same kind of preprocessing can be applied to an arbitrary text. Moreover, tagged protein names in the text may include more than one word; in order to treat them as a single entity in further processing stages, we replace them in the same manner as formulas. Finally, we take all possible pairwise combinations of proteins in each sentence and create one sentence for each combination where only this combination is tagged. Part of this process is shown in Example 2.

Example 2 After preprocessing, the second sentence from Example 1 will look as follows (only two instances out of 10 are presented for the sake of conciseness):

[4]As the focus of this paper is on the mining of interactions, we assume that protein name recognition has already taken place. For the recognition of the protein names themselves we refer to [3, 7, 17].

A low GerE concentration, which was observed during the experiment, activated the transcription of cotB by final sigmaK RNA polymerase, whereas a higher concentration was needed to activate transcription of cotX or cotC.

. . .

A low GerE concentration, which was observed during the experiment, activated the transcription of cotB by final sigmaK RNA polymerase, whereas a higher concentration was needed to activate transcription of cotX or cotC.

3.2 Feature vector construction

After the text preprocessing stage, for each sentence we build a feature vector that summarizes important syntactic information on the parse tree and the typed dependency tree, which are both ways of representing sentence structure. A parse tree is a tree (in terms of graph theory) that represents the syntactical structure of a sentence. Words from the sentence are leaves of the parse tree and syntactical roles are intermediate nodes, so a parse tree represents the nesting structure of multi-word constituents. A dependency tree on the other hand represents interconnections between individual words of the sentence. Hence, all nodes of the dependency tree are words of the sentence, and edges between nodes represent syntactic dependencies. In typed dependency trees, edges are labeled with syntactic functions (e.g., subj, obj). Figure 1 depicts the typed dependency tree and parse tree for the first sentence of Example 1.

During the feature extraction phase, we parse each sentence with the Stanford Parser[5]. For each tagged pair of proteins (recall that after the preprocessing step each sentence has only one such pair), we extract a linked chain [16] from the dependency tree. The dependency tree is unordered w.r.t. the order of the words in the sentence; hence to produce patterns uniformly, we order the branches in the linked chain based on the position of the words in the initial sentence. Thus the left branch contains the words that occur earlier in the sentence and the right branch the words that occurs later. The absolute majority of the branches in the linked chains from the datasets we examined contain no more than 6 edges, and those which contain more are negative instances, so we choose feature vectors with 6 features for each branch to cover all positive examples from the training set. Therefore, we use the first 6 dependencies from the left branch as the first 6 features in the feature vector. Likewise, the first 6 dependencies from the right branch correspond to features 7 through 12. Moreover, to better describe the structure of the relation we incorporate information from the parse tree as well, namely the length of the path from the root of the parse tree to each protein as the 13th and the 14th feature, and the number of nested subsentences in these paths as the 15th and the 16th feature.

Example 3 Below is the feature vector of the first sentence from Example 1:

nsubj	*nn*					(left branch)

[5]`http://nlp.stanford.edu/downloads/lex-parser.shtml`. We have used the standard version trained on the WSJ corpus.

dobj	*prep_of*					(right branch)

5	7	0	0	(parse tree information)

To build this feature vector, we extracted a linked chain between the two proteins *Sigma F* and *Sigma E*, as shown in Figure 1. It is already ordered, i.e., *Sigma F* precedes *Sigma E* in the sentence, so we do not need to swap these branches. As the left branch contains only 2 dependencies — *nsubj* and *nn*, 4 slots in the vector remain empty. Features 7-12 for the right branch are filled in the same manner. *Sigma F* is at depth 5 in the parse tree while *Sigma E* is at depth 7, and the parse tree in Figure 1 does not contain subsentences. All this information is reflected in the last 4 features of the vector. Note that the resulting feature vector contains several empty fields; only the most complicated sentences will have a value for each feature in the vector.

3.3 Training a classifier

We call an interaction *straight* if the agent appears earlier than the target in the sentence, otherwise we call it *inverse*. When presented with a feature vector representing a sentence, the classifier should be able to detect whether the sentence (1) describes a straight interaction, (2) an inverse interaction, or (3) no interaction at all.

To build the classifier, we compare a decision tree algorithm (C4.5 implementation [15]) and the BayesNet classifier [10]. These two algorithms represent classical instances of two branches of machine learning — rule induction and statistical learning — which employ different approaches to data processing. Decision trees consist of internal nodes which represent conditions on feature values, and leaves which represent classification decisions that conform to the feature values in the nodes on the way to this leaf. The BayesNet classifier on the other hand is represented as a directed graph with a probability distribution for each feature in the nodes and with the edges denoting conditional dependencies between different features. When we use the BayesNet classifier we apply a conditional independence assumption, which means that probabilities of node values depend only on probabilities of values of their immediate parents, and do not depend on higher ancestors. This corresponds to the reasonable assumption that the syntactic role of a node in the linked chain depends on the syntactic role of its parent only.

To overcome the problem of missing values (which occur frequently in the feature vectors), in the BayesNet classifier we replace them by a default value. With C4.5, to classify an instance that has a missing value for a given node, the instance is weighted proportionally to the number of instances that go down to each branch, and recursively processed on each child node w.r.t. the assigned weight. This process is described in more detail in [18].

4. DATASETS

To evaluate the performance we used two datasets. The AImed dataset [1] is compiled from 197 abstracts extracted from the Database of Interacting Proteins (DIP) and 28 abstracts which contain protein names but do not contain interactions. Since we are interested in retrieving protein interactions, in this paper we use only the former set of abstracts. The connection between a full name of a protein and

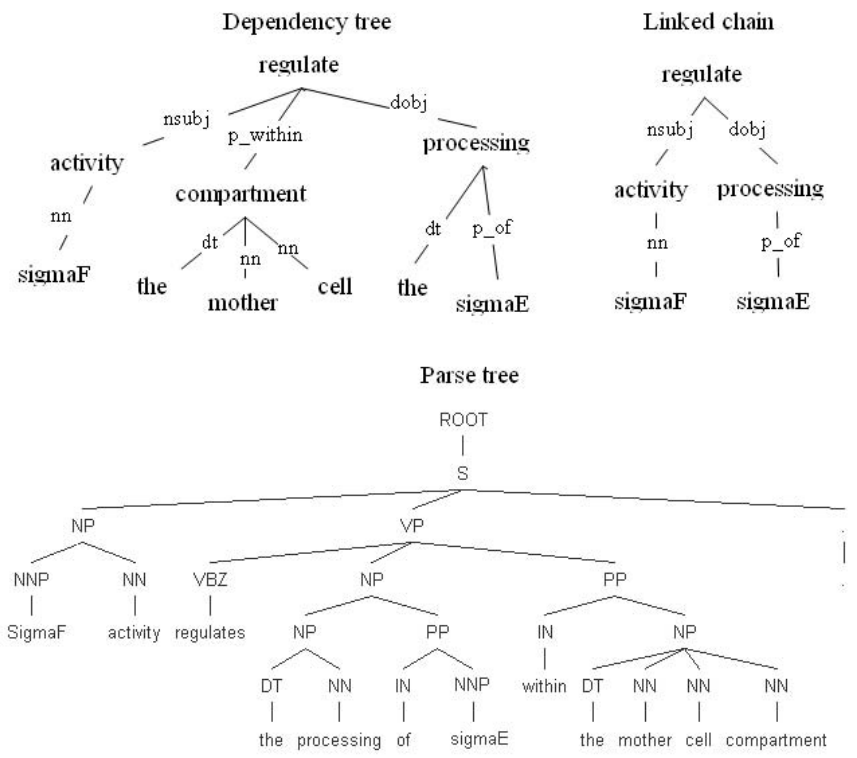

Figure 1: Dependency tree, parse tree, and linked chain for the first sentence of Ex. 1.

its abbreviation, e.g. *tumor necrosis factor (TNF)*, is annotated as an interaction in the AImed dataset. Since such an annotation is not concerned with an actual interaction between different proteins, we omit this kind of data from our experiments. Furthermore, we remove nested protein annotations, which wrap around another protein or interaction annotation. Finally, TI- and AD- sections as well as PG- prefixes, which are Medline artifacts, are removed. The AImed dataset does not contain annotations for agent and target proteins, hence we cannot use it to evaluate the performance of our role detection algorithm. However, we can still use it to assess the quality of our algorithm for the detection of protein interactions in general. Note that in this case the three-class classification problem described in Section 3 collapses to a two-class problem, i.e., detecting whether the sentence under consideration (1) describes an interaction, or (2) does not.

The second dataset [12] originates from the Learning Language in Logic (LLL) relation mining challenge on Genic Interaction Extraction[6]. This dataset contains annotated protein/gene interactions concerned with *Basilicus subtilis* transcription. The sentences in the dataset do not make up a full text, but they are individual sentences taken from several abstracts retrieved from Medline. Interactions in this dataset are asymmetric, and agent and target roles are annotated for each interaction. This gives us the opportunity to evaluate our approach for the role labeling task as well. In the experiments, we consider both the two-class and the three-class classification problem for the LLL dataset.

[6]`http://genome.jouy.inra.fr/texte/LLLchallenge/`

Dataset	# sentences	# pos. inst.	# neg. inst.
AImed	1978	816	3204
LLL'05	77	152	233

Table 1: Datasets used in the experiments

More information about the datasets is listed in Table 1. From this table, it is clear that the AImed dataset is highly imbalanced, as there is a strong bias to negative examples. To the best of our knowledge, these are the only two publicly available datasets containing annotations of protein interactions and hence suitable to evaluate our approach.

5. EXPERIMENTAL EVALUATION

For evaluating our approach, we set up two kinds of experiments depending on whether protein role information was taken into account. For detecting symmetric interactions, we used 10-fold cross validation for both the AImed and the LLL05 dataset, and we also ran experiments with AImed as training set and LLL05 as test set. For directed interaction detection, 10-fold cross validation was done on the LLL dataset only, since AImed does not contain role information. All experiments use Weka [18] for implementing machine learning methods.

The difference in the datasets requires different parameters to achieve optimal performance. As we have mentioned, the AImed dataset is imbalanced and using it for training tends to lead to a bias towards classifying examples as negative (independently of the training scheme). For this reason, we use cost-sensitive learning [18] to decrease the bias when AImed is used as a training set. Moreover, in the C4.5 implementation for the AImed dataset, we build a binary decision tree, i.e., at each node the algorithm tests only one value of one feature. Otherwise, the algorithm would decide that the empty tree classifies the dataset in the best way, and all examples would be classified as negative (again, because of the biased dataset).

The first three pictures in Figure 2 depict the performance of our approach for symmetric relations and its comparison with state-of-the-art approaches evaluated on the same datasets. To study the trade-off between recall and precision, we use a confidence threshold p between 0 and 1 such that an instance (a feature vector) is retrieved iff the classifier has a confidence of at least p that the instance describes a real protein interaction. The BayesNet classifier provides such a confidence value naturally, because its output is a class distribution probability for each instance. Decision trees can also be easily adapted to produce a probabilistic output by counting training examples at the leaf nodes. If a sentence that is being classified ends up at a leaf node, the confidence of the classifier that it is a positive instance, is the proportion of positive training examples to all training examples at that leaf node. Decreasing the threshold p from 1 to 0 allows to increase the recall at the cost of a drop in precision.

For the LLL dataset (whose interactions are treated as symmetric in this case), we provide a comparison with the RelEx system [8], depicted by a $*$ in the first picture. In the second picture, we compare our approach on the AIMed dataset to the PAS-approach reported in [19]. For the cross-dataset experiment, the third picture compares it with the subsequence kernel method from [2]. It is clear from these pictures that overall our method obtains similar or even better results than the existing approaches. A detailed analysis of these results can be found in [6].

To study the classification power of grammatical relations for role detection, we have performed a cross-validation experiment for the directed LLL dataset. Since in this case, instances are classified into three classes instead of two, we should adapt the evaluation technique to provide results that are comparable to the previous experiments. In particular, we reduce the results to two-class classification. Depending on which aspect we wish to evaluate, this transformation can be done in three ways: *straight* class against others, *inverse* class against others, and *straight* and *inverse* classes combined against *'no interaction'* class. To the best of our knowledge, so far nobody has provided a separate analysis for *straight* and *inverse* interactions, hence we provide a comparison with other approaches only for the combined classification evaluation.

In order to understand how the evaluation was done, recall that every classified instance has three probability values, denoting the probability of an instance to belong to each of the three classes. First, to evaluate the quality of the *straight* interaction extraction, we varied a confidence threshold for *straight* interactions and consider only the probability of the instance to be classified as *straight* w.r.t. the confidence threshold. The middle right picture of Figure 2 illustrates the fact that C4.5 obtains substantially better results than BayesNet for this type of interactions. Secondly, we evaluate the quality of the *inverse* interaction extraction, using the same methodology as for the *straight* class. In the bottom left picture, we can observe that in this case BayesNet performs much better, keeping the precision rate around 80% until recall reaches almost 90%. Finally, we evaluated the results for the combined interaction class. Here, we applied the confidence treshold to the negative (non-interacting) class probability, and for each instance above the treshold we chose the class with the highest probability (*straight* or *inverse*). The bottom right picture shows the recall-precision curves for this type of evaluation, which we compare with the performance of the RelEx approach depicted by $*$, and that of the approach described in [11] and depicted by Δ. The latter approach uses dependency tree levels to construct a keywords-based feature vector, and then uses the vectors to train a probabilistic classifier. Note that the decision tree classifier approaches a RelEx result (similarly as in the top left picture), while both classifiers outperform the keywords-based machine learning technique (dependency tree levels).

It is also interesting to consider the confusion matrices associated to the LLL cross-validation experiment, shown in Table 2. The columns of a matrix represent the instances in a predicted class, while the rows show the amount of instances in an actual class. For example, the value 19 in the first column of the C4.5 results in Table 2 represents the number of instances containing *straight* interactions that were classified as non-interacting. Analysis of the matrices reveals that the confusion (mislabeling) between the two types of interactions is rather small for both classifiers — 5 instances for each interaction type for C4.5, and 6 and 3 instances for BayesNet classifier. This means, that grammatical roles give us a good clue not only to detect the interaction itself, but also to determine the protein role in interaction.

C4.5 results

none	direct	inverse	<-prediction
204	10	15	none
19	51	5	straight
22	5	44	inverse

BayesNet results

none	direct	inverse	<-prediction
205	11	13	none
30	42	3	straight
19	6	46	inverse

Table 2: Confusion matrices for directed LLL experiment

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented an approach to protein interaction mining that relies exclusively on grammatical relations and parsing information, and hence is fully context independent. In this way, our method is different from existing approaches for protein interaction detection that typically rely on shallow parsing information, sometimes augmented with full parsing information. More in particular, we proposed a clean and generally applicable approach in which for each sentence a feature vector is constructed that contains 12 features with information on the dependency tree and 4 features with information on the parse tree of the sentence.

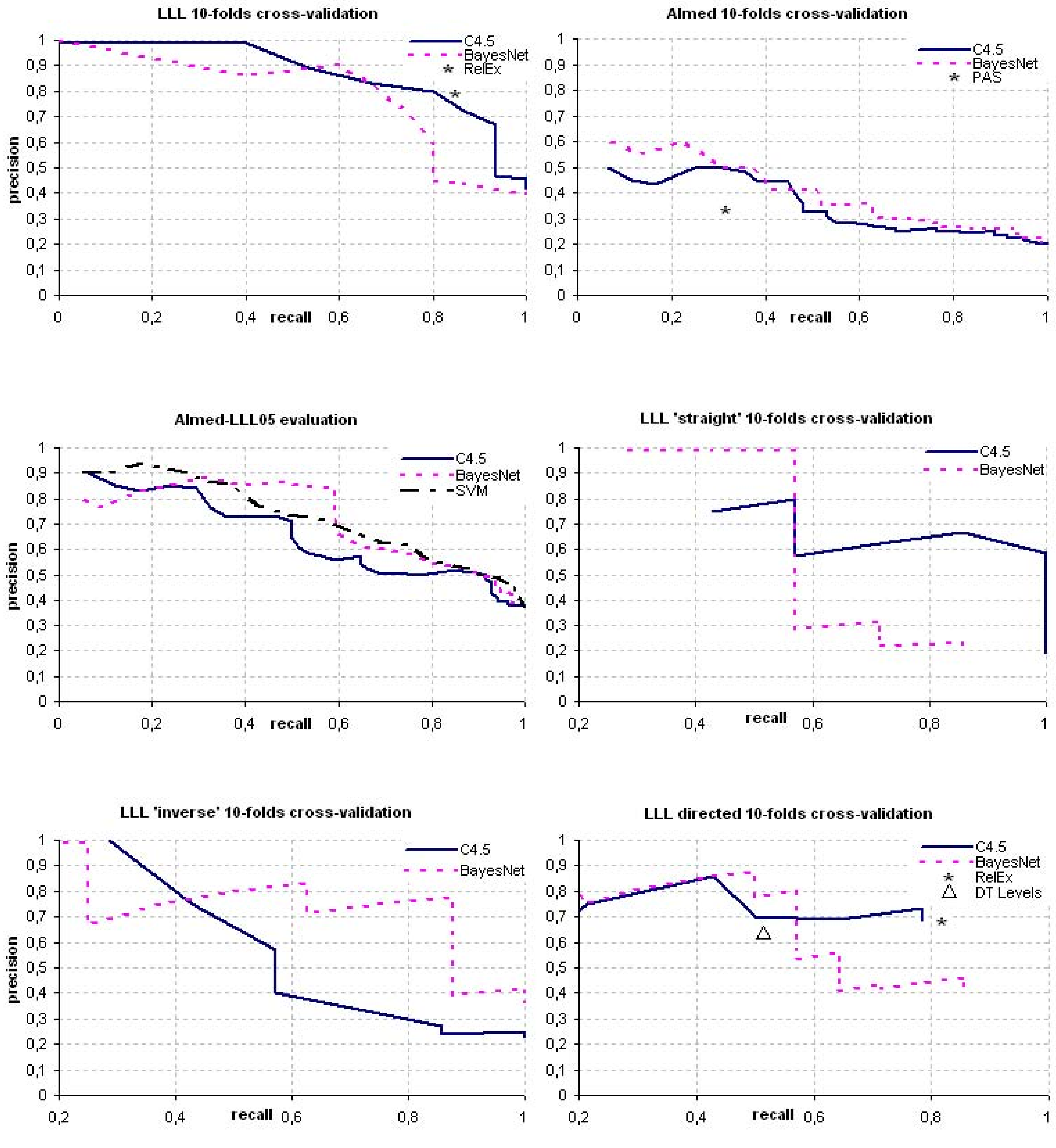

Figure 2: Recall-precision charts

Next, we fed these feature vectors as inputs to a C4.5 and a BayesNet classifier, as representatives of rule induction and statistical learning algorithms, respectively.

Using these standard data mining algorithms and no shallow parsing or lexical information whatsoever, we were able to obtain results which are comparable with state-of-the-art approaches for protein relation mining. This result is promising since a method that uses only full parsing information does not depend on specific interaction keywords and is less affected by the size and/or the heterogeneity of the training corpus. Moreover, we applied our approach to the more complicated task of directed interaction extraction that requires the detection of interaction participant's roles. Cross-validation evaluation showed results outperforming traditional keyword-based approaches. Another interesting finding is that the role confusion in the detected interactions was very low, which supports the assumption that grammatical relations give important clues for role detection.

As this paper presents work in progress, quite some ground remains to be covered, including a more complete comparison with existing methods. Among other things, it would be interesting to build an SVM model with our feature vectors and compare the results with those of shallow and combined parsing based approaches that rely on kernel methods as well. A final intriguing question is whether an augmentation with shallow parsing information could increase the performance of our approach.

Acknowledgement
Chris Cornelis would like to thank the Research Foundation–Flanders for funding his research.

7. REFERENCES

[1] R. Bunescu, R. Ge, J.R. Kate, M.E. Marcotte, R.J. Mooney, K.A. Ramani and W.Y.Wong *Comparative Experiments on Learning Information Extractors for Proteins and their Interactions*, Artificial Intelligence in Medicine 33(2), 2005.

[2] R.Bunescu and J.R.Mooney *Subsequence Kernels for Relation Extraction*, in: Proc. 19th Conf. on Neural Information Processing Systems (NIPS), 2005.

[3] N.Collier, C.Nobata and J.Tsujii. *Extracting the Names of Genes and Gene Products with a Hidden Markov Model*, in: Proc. 17th Int. Conf. on Computational Linguistics, 2000.

[4] G. Erkan, A. Ozgur and D.R. Radev *Extracting Interacting Protein Pairs and Evidence Sentences by using Dependency Parsing and Machine Learning Techniques*, in: Proc. 2nd BioCreAtIvE Challenge Workshop — Critical Assessment of Information Extraction in Molecular Biology, 2007.

[5] C. Giuliano, A. Lavelli and L. Romano *Exploiting Shallow Linguistic Information for Relation Extraction From Biomedical Literature*, in: Proc. 11th Conf. of the European Chapter of the Association for Computational Linguistics (EACL 2006).

[6] T.Fayruzov, M.De Cock, C.Cornelis, V.Hoste *DEEPER: a Full Parsing based Approach to Protein Relation Extraction*, in: Proc. 6th European Conference, EvoBIO 2008.

[7] K.Fukuda, A.Tamura, T.Tsunoda, T.Takagi *Toward Information Extraction: Identifying Protein Names from Biomedical Papers*, Proc. Pacific Symp. on Biocomputing, 1998.

[8] K. Fundel, R. Küffner, and K. Zimmer *RelEx—Relation extraction using dependency parse trees*, Bioinformatics 23(3) 2007.

[9] J. Hakenberg, C. Plake, U. Leser, H. Kirsch and D. Rebholz-Schuhmann *LLL'05 Challenge: Genic Interaction Extraction — Identification of Language Patterns Based on Alignment and Finite State Automata*, in: Proc. ICML05 Workshop: Learning Language in Logic, 2005.

[10] F.V. Jensen. *An Introduction to Bayesian Networks* Springer, 1996.

[11] S. Katrenko and P. Adriaans *Learning Relations from Biomedical Corpora Using Dependency Tree Levels*, in: Proc. BENELEARN conference, 2006

[12] C. Nédellec *Learning Language in Logic - Genic Interaction Extraction Challenge*, Proc. ICML05 Workshop: Learning Language in Logic.

[13] T. Ono, H. Hishigaki, A. Tanigami and T. Takagi *Automated Extraction of Information on Protein-Protein Interactions from the Biological Literature*, Bioinformatics, 17(2) 2001.

[14] J. Pustejovsky, J. Castano, J. Zhang, M. Kotecki, B. Cochran *Robust relational parsing over biomedical literature: Extracting inhibit relations*, in: Proc. of the 7th Pacific Symp. on Biocomputing, 2002.

[15] J.R. Quinlan. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, 1993.

[16] M. Stevenson and M.A. Greenwood. *Comparing Information Extraction Pattern Models*, in: Proc. IE Beyond The Document Workshop (COLING/ACL 2006).

[17] L.Tanabe and W.J.Wilbur. *Tagging Gene and Protein Names in Biomedical Text*, Bioinformatics, 18(8), 2002.

[18] I.H. Witten and E. Frank *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[19] A. Yakushiji, Y. Miyao, Y. Tateisi and J. Tsujii *Biomedical Information Extraction with Predicate-Argument Structure Patterns*, in: Proc. 1st Int. Symp. on Semantic Mining in Biomedicine, 2005.