A closer look at learning relations from text
Katrenko, S.

Citation for published version (APA):
Katrenko, S. (2009). A closer look at learning relations from text

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
FEATURE ENGINEERING USING SYNTACTIC STRUCTURES

ABSTRACT

In this chapter we address the relation learning problem by exploring dependency structure. We propose a novel representation which takes into account syntactic information and allows for using different machine learning methods. To carry out syntactic analysis, three parsers, LinkParser, Minipar and Charniak parser were used. Our experimental findings suggest that the final performance depends on the parser that is being used. In addition, we have studied the impact of ensemble methods on learning relations using the representation we proposed. Given that recall is very important for relation learning, we explore ways of improving it. We demonstrate that ensemble methods provide higher recall and precision than individual classifiers alone. A part of this chapter was published in KDECB (Knowledge Discovery and Emergent Complexity in BioInformatics), Lecture Notes in Bioinformatics (LNBI), vol. 4366 authored by Sophia Katrenko and Pieter Adriaans, “Learning Relations from Biomedical Corpora Using Dependency Trees”.

5.1 INTRODUCTION

We have already shown in the previous chapters that automatic relation recognition is important for various applications and in a variety of fields. This chapter focuses on exploring syntactic information that can be found in dependency structures. More precisely, we aim at a flexible representation of relation instances such that it will allow us to use different machine learning methods and still be sufficient for accurate extraction. Our experiments are conducted in the biomedical domain.

The biggest collection of medical documents is Medline, with 2,000 citations added every week. The large size of this collection makes it impossible to annotate it all by humans. Consequently, there have been several attempts to create smaller annotated corpora based on Medline, such as Genetag used for gene/protein named entity recognition (NER) [184], or MedTag, the corpus comprising Genetag, MedPost and ABGene [175]. There have also been corpora created with a special purpose to be used by various challenges, e.g. a corpus of annotated gene-protein relations for the "Genic Interaction Extraction Challenge" [136] or corpora used for the BioCreAtive competition [93]. Boosted interest in data mining from biomedical data can be explained by the large amount of
work that is being done in the biomedical field. Extracting of named entities and relations is the first step in building repositories, ontologies and other resources that would allow integration of already available knowledge with information found in text.

The chapter is organized as follows. We start by studying how relations are realized syntactically and provide motivation for a new feature-value representation. In the next sections various machine learning methods (base and ensemble classifiers) are used on two biomedical data sets for interaction extraction. We also consider several parsers and study whether the choice of syntactic parser influences the final performance.

5.2 SYNTACTIC INFORMATION FOR RELATION LEARNING

The approach we take follows the definition of relation discovery as a two-step process, concentrating on the second step only. We assume that we have already identified the arguments of a relation. In what follows, we present our method and give a motivation from both linguistic and machine learning perspectives.

5.2.1 Syntactic realization of relations

In the linguistic tradition, a syntactic structure of a sentence can be presented either by constituency or by dependency analysis [149]. The main distinction between the two approaches is the following. By constituency analysis a sentence is represented by non-overlapping groups of words, whereas dependency is a hierarchical relation where every word in a sentence is linked to a word dominating it. For the sentence in (5.1), the constituency analysis is given in (5.2) and the dependency structure is presented in Fig. 6. In (5.2), all words are grouped into noun phrases (NP), verb phrases (VP), prepositional phrases (PP) and pre-verbal adverb phrases (ADVP). Note that one phrase can contain other phrases, but they never overlap.

(5.1) Cdc25 can be activated in vitro in a Raf1-dependent manner.

(5.2) (S (NP Cdc25) (VP can) (VP be (VP activated (ADVP in vitro) (PP in (NP a Raf1-dependent manner))))) (. .))

Unlike constituency analysis, dependency structure does not imply linear ordering and in our view is more appropriate for the relation learning task. At closer inspection of the sentence (5.1), it can be found that Cdc25 and Raf1 are interacting proteins. The evidence supporting such claim is the word activated, so it is possible to rephrase the sentence as ‘Raf1 activates Cdc25’. It is, however, difficult to use the constituency structure to detect this relation automatically. In contrast, the root of the
dependency tree depicted in Fig. 6 already consists of the word *activated*. It is also given that *Cdc25* is a subject of the sentence and that it is in passive voice. Following such analysis, it becomes clear that *Cdc25* is an argument of the binary relation of activation. Assuming activation to be an asymmetric relation, we can conclude that *Cdc25* is a target and *Raf1* is an agent of this relation, or *activation(Raf1,Cdc25)*.

![Dependency tree](image)

Figure 6: The dependency structure for Example (5.1)

Sentences in the biomedical domain are often complex and contain various linguistic phenomena such as conjunction, enumeration, and negation. To estimate whether these phenomena occur frequently in text data, we inspected the sentences used by the LLL challenge [136]. It turned out that 24.68% of all sentences contain either conjunctions or enumerations or both. We believe that such phenomena are covered well if the syntactic structure is used. Consider, for example, the sentence in (5.3):

Conjunctions
The expression of rsfA is under the control of both sigma(F) and sigma(G).

There are two interactions expressed in this sentence, a relation between sigma(F) and rsfA on the one hand, and between sigma(G) and rsfA, on the other. If one uses a local context only, it will most likely miss some of the interactions, while the dependency paths between the arguments provide necessary and (hopefully) sufficient information to extract all relation mentions. The dependency tree for this sentence will show that sigma(F) and sigma(G) are siblings whose parent is a node labeled by control. Further, syntactic functions encountered when traversing a tree (such as subj (subject), comp_of (complement) or comp_under (complement)) contribute to relation extraction as well. Using them would enable one to distinguish between passive and active voice which becomes important when one wishes to identify a direction of interaction (i.e., 'X activates Y' vs. 'X is activated by Y').

Conjunction and disjunctions are certainly not the only phenomena which need to be taken into account. The other very common phenomenon is enumeration, as shown in (5.4).

(5.4) These results suggest that YfhP may act as a negative regulator for the transcription of yfhQ, yfhR, sspE and yfhP.

This example contains four interactions between a protein (YfhP) and four genes (yfhQ, yfhR, sspE and yfhP). Again, by exploring dependency structure we may notice that all gene names fall under a node transcription. This means that all four relation instances can be represented in a similar way. Moreover, if one of them is correctly identified as a positive instance of interaction, the other three will be classified as positive instances as well.

Generally speaking, the root of a dependency tree is often a verb. As shown by Sekimizu et al. [170], a relation between two or more arguments is also often expressed by verbs. We can therefore conclude that the root of a dependency tree conveys information crucial for relation learning. Furthermore, by examining the parent and children of a given node, one can notice that they constitute a local context important for argument identification. In Example (5.1) dependent, which is a parent of the node labeled by Raf1 indicates that Raf1 is an agent and not a target of the interaction.

To give further motivation for the levels selection, we examined the roots of the syntactic trees from the LLL data set. This data set was also processed by 2 researchers in the bioinformatics field whose main intention was to detect all verbs used to express interactions between two biomolecular instances. Such information enables further analysis of the tree level. Namely, it is possible to compare the list of interaction verbs
against the words found in the root of the dependency structures. Fig. 13 in Appendix of this chapter presents the distribution of verbs selected by our experts which were found in the root of a sentence. Note that some verbs (e.g. to prevent) could not be found in the root which leads us to the conclusion that other levels in the dependency structures should be considered as well.

50 out of 77 sentences in the LLL training set contain the selected verbs as roots of the dependency structures. 74.07% of all selected verbs can be found in a root of a dependency tree. Moreover, the verbs which do not occur in a root of a tree, can still be found on other levels. For instance, although stimulate and rely are not present in a root, they occur in the parents’ nodes of the arguments of a relation, as in Example (5.5).

(5.5) During endospore formation in Bacillus subtilis, the DNA binding protein GerE stimulates transcription from several promoters that are used by RNA polymerase containing sigmaK.

There are several advantages to considering dependency tree levels. First of all, it is possible to test our hypothesis in order to discover which levels are the most important for relation learning. Selecting tree levels can be considered as a feature engineering step. Moreover, since the final representation is of the attribute-value type, it is possible to test different machine learning methods. It is of considerable interest to apply ensemble methods [41]. Experiments for the named entity recognition task have already demonstrated that the use of meta-learning improves the accuracy of classification [165].

5.2.2 Defining levels

We divide all features into two groups, local and global context. To reduce data sparseness, we decided to use lemmata instead of words. A parent of a given node (P) and its two children form a local context. The features of a parent and a child are lemmata and the syntactic function between a node in question and a parent (a child). Since a tree is an acyclic graph, each node has at most one parent but can have more than one child. We limited ourselves to two children, C₁ and C₂.

In addition to a root of a tree (R), a global context consists of a least common subsumer (LCS).

Definition 12 (Least common subsumer (LCS)). Given two nodes A and B in a dependency tree T, a least common subsumer LCS(A, B) is a node L, such that L is ancestor for both A and B, and there exist no other node N being an ancestor for A and B, such that L is ancestor of N. There is exactly one LCS for any two nodes in a dependency tree.
For example, for the words *a* and *Raf* in Fig. 6, the least common subsumer is *manner*. Although such nodes as *in vitro* and *activated* are all ancestors of *a* and *Raf*, they are not least common subsumers.

The motivation to include the least common subsumer in the feature set comes from the observation that the arguments of the relation can be located closer to the leaves of a tree and a root in such cases is not sufficiently discriminative. Consider, for instance, the sentence in (5.6). Here, there are two relation mentions, namely interaction(*cwI*H, *sigma*(*H*)) and interaction(*gerE*, *cwI*H). The root of the dependency tree for this sentence is *depended*, which indicates an interaction between *gerE* and *cwI*H. However, it is not sufficient to discover a relation between *gerE* and *sigma*(*H*), it is only possible to extract it if a least common subsumer (*dependent*) is used.

(5.6) Expression of the sigma(K)-dependent cwI*H* gene depended on *gerE*.

### Feature sets

Some parsers treat a subordinate clause as separate producing not a single tree for a sentence but two. We decided therefore to define two features, one for the root of the first argument (*R*1) and second for the root of the second argument(*R*2). Table 9 illustrates how the features have been grouped into feature sets given two nodes X and Y and the relation *R*(X, Y). The lower indices in Table 9 correspond to two arguments, X and Y.

<table>
<thead>
<tr>
<th>Feature set (FS)</th>
<th>Features</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS1</td>
<td>LCS</td>
<td>a least common subsumer</td>
</tr>
<tr>
<td>FS2</td>
<td><em>C</em>&lt;sub&gt;1&lt;/sub&gt;_X, <em>C</em>&lt;sub&gt;2&lt;/sub&gt;_X, <em>C</em>&lt;sub&gt;1&lt;/sub&gt;_Y, <em>C</em>&lt;sub&gt;2&lt;/sub&gt;_Y, LCS</td>
<td>2 children nodes per relation argument and a least common subsumer</td>
</tr>
<tr>
<td>FS3</td>
<td><em>P</em>&lt;sub&gt;X&lt;/sub&gt;, <em>P</em>&lt;sub&gt;Y&lt;/sub&gt;, LCS</td>
<td>2 parent nodes per relation argument and least common subsumer</td>
</tr>
<tr>
<td>FS4</td>
<td><em>C</em>&lt;sub&gt;1&lt;/sub&gt;_X, <em>C</em>&lt;sub&gt;1&lt;/sub&gt;_Y, <em>P</em>&lt;sub&gt;X&lt;/sub&gt;, <em>P</em>&lt;sub&gt;Y&lt;/sub&gt;, LCS, <em>R</em>&lt;sub&gt;X&lt;/sub&gt;</td>
<td>2 children nodes and parent nodes per relation argument, a least common subsumer and a root node</td>
</tr>
<tr>
<td>FS5</td>
<td><em>C</em>&lt;sub&gt;1&lt;/sub&gt;_X, <em>C</em>&lt;sub&gt;2&lt;/sub&gt;_X, <em>C</em>&lt;sub&gt;1&lt;/sub&gt;_Y, <em>C</em>&lt;sub&gt;2&lt;/sub&gt;_Y, <em>P</em>&lt;sub&gt;X&lt;/sub&gt;, <em>P</em>&lt;sub&gt;Y&lt;/sub&gt;, LCS, <em>R</em>&lt;sub&gt;X&lt;/sub&gt;, <em>R</em>&lt;sub&gt;Y&lt;/sub&gt;</td>
<td>2 children nodes and a parent node per relation argument, a least common subsumer and roots</td>
</tr>
</tbody>
</table>

Table 9: Composed feature sets
When considering the fifth dataset (FS5), the example in Fig. 6 can be represented as in Table 10:

<table>
<thead>
<tr>
<th>feature</th>
<th>‘Cdc25’</th>
<th>‘Raf1’</th>
</tr>
</thead>
<tbody>
<tr>
<td>C¹</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C²</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>P</td>
<td>activate_s</td>
<td>dependent_lex-mod</td>
</tr>
<tr>
<td>LCS</td>
<td>activate</td>
<td>activate</td>
</tr>
</tbody>
</table>

Table 10: Illustrating features for Example 5.1

Note that we incorporated the syntactic labels into the parent-features P. ‘Cdc25’ is linked to the word ‘activated’ by the syntactic function ‘s’ (which stands for a subject), while ‘Raf1’ is connected to ‘dependent’ by ‘lex-mod’ (which stands for a modifier).

5.3 EXPERIMENTAL SET-UP

5.3.1 Datasets

In our experiments, we have used two data sets. One of them is AImed [19] and the other is a data set created within the “Genic Interaction Extraction” challenge [136] (from now, we refer to it as the LLL (Learning Language in Logic) data set). The AImed data set consists of examples of protein-protein interactions. It has been compiled from 225 Medline abstracts and annotated by experts. The second data set, LLL, has been created by extracting Medline abstracts on Bacillus subtilis. It also includes annotations created by experts, with the distinction that the focus is on interactions between genes and proteins. The LLL data set consists of 77 sentences and 165 annotated interactions. More statistics on the data sets are given in Table 51 in the Appendix of this thesis.

5.3.2 Data preprocessing

The LLL data set already consists of the tokenized sentences accompanied by syntactic analysis. For parsing, the LLL organizers have used the LinkParser whose output has been verified by experts. Besides this, a dictionary of genes and proteins has been provided so we annotated all occurrences of the dictionary items in the text as biological entities (in the dictionary, no distinction between genes and proteins has been made).

The AImed data set has been preprocessed by us. We have used a tokenizer based on white spaces. We have also found that the present...
annotation sometimes contains protein tags surrounding the interaction tags as shown in Example 5.7. Here, Ras is annotated as a protein being an argument of an interaction with RIN1. In addition, Ras binding protein is also annotated as a protein. As explained below, we have constructed false interactions for training purposes based on the entities annotated as proteins and not being part of a relation. We scanned the AImed data set and found 14 cases where the annotation of a protein included annotation of interactions as its part. While carrying out preprocessing, the external protein tags have been removed.

5.7 Human ⟨p1 pair="1"⟩⟨prot⟩RIN1 ⟨/prot⟩⟨/p1⟩ was first characterized as a ⟨prot⟩⟨p2 pair="1"⟩⟨prot⟩Ras ⟨/prot⟩⟨/p2⟩ binding protein ⟨/prot⟩ based on the properties of its carboxyl-terminal domain.

The data sets we have used provide annotations of the binary interaction relation. The arguments of a relation in LLL always occur in the same sentence, while in AImed they might be in adjacent sentences. We discarded all examples from the latter corpus that were spanning over several sentences.

In order to obtain the negative interactions, we have followed the closed world assumption. However, it has been used in a different way for each of the data sets. The interactions between proteins in AImed are considered to be symmetric. Therefore, the false positives are created as all pairs of proteins being not arguments of the interaction relation as well as pairs where one of the proteins is an argument of a relation in a given sentence. LLL data set contains interactions between genes and proteins which are treated as instances of an asymmetric relation. Because of this, the false interactions are produced as pairs of biomolecular entities (i.e., proteins or genes) which do not participate in a relation but also those where the arguments of a relation are flipped (e.g., a pair (X,Y) where X and Y are biomolecular entities will be considered a false positive for the true interaction (Y,X)).

After constructing the training set, we received 909 training instances for the LLL data set, 165 of which were positive examples. For the AImed corpus, we obtained 5,106 instances with 1,006 of them being positive examples.

5.3.3 Parsers

Syntactic analysis is an important step for many text mining tasks in the biomedical domain. In most cases, moving to another domain requires a parser adaptation. Such need is motivated by domain-specific vocabulary and different stylistic peculiarities [99]. Lease and Charniak [99] studied parsing in the biomedical domain and showed that by adapting the Charniak parser, the error rate decreases in 14.2%. Since our approach
also crucially depends on the quality of the syntactic analysis, we selected several state-of-the-art parsers to experiment with.

The first parser is LinkParser which has been used by the organizers of the LLL challenge. It produces relations between pair of words in a sentence, linked by a syntactic function, such as subject or complement. This parser has been adapted to the biomedical domain at MIG Lab. Moreover, the output of the parser was verified by hand, so it can be referred to as the gold standard analysis.

Minipar is another parser, which is freely available. On the Susanne corpus, it achieves 88% precision and 80% recall. Unlike LinkParser, Minipar has not been trained on biomedical corpora and it needs to be investigated whether it can be successfully applied to a given task.

The third syntactic analyzer that we considered is the Charniak parser, whose output can be transformed to dependencies between pairs of words. This statistical parser was trained on the Penn BioIE treebank and part of the Genia treebank and reaches 85% PARSEVAL F-measure when performing 10-fold cross-validation. However, in this case the dependencies do not contain syntactic functions, as in the output of the LinkParser. We present an example of the syntactic analysis by all three parsers in Appendix I of this chapter (Fig. 10, Fig. 11, and Fig. 12).

5.4 RESULTS AND DISCUSSION

The results we present below have been received by 10-fold cross-validation for the AIMed data set and for the LLL data set. We have used the implementation of the machine learning methods from the Weka toolkit [204].

5.4.1 Impact of the pre-defined levels

We studied the impact of the feature sets mentioned above on recall and precision. First, we started with a feature set containing the least common subsumer only (FS1). In our view, this feature set is similar to the pattern approaches whose main objective is to find a link (a so-called relation word) between two arguments.

As Table 11 suggests, the least common subsumer often includes the words important for relation learning in the biomedical domain. In the list below, such words are written in italic. Comparing this list against the list of verbs identifying relations [170], we found that they significantly overlap.

However, using this feature set provides recall and precision, which can likely be further improved. This supports our hypothesis about precompiled list of patterns used for relation extraction - in most cases, they cannot cover unseen data well. Our results are in line with those reported by Ahmed et al. [4].
The second feature set, FS₂, consists of FS₁ and the children of the two arguments. As the results in Fig. 7a and in Fig. 7b suggest, recall already increases by adding the information about the children.

<table>
<thead>
<tr>
<th>Words (LCS)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>to be</td>
<td>449</td>
</tr>
<tr>
<td>to bind</td>
<td>211</td>
</tr>
<tr>
<td>to interact</td>
<td>182</td>
</tr>
<tr>
<td>to inhibit</td>
<td>65</td>
</tr>
<tr>
<td>to associate with</td>
<td>47</td>
</tr>
<tr>
<td>to contain</td>
<td>47</td>
</tr>
<tr>
<td>to reveal</td>
<td>43</td>
</tr>
<tr>
<td>to induce</td>
<td>42</td>
</tr>
<tr>
<td>to include</td>
<td>39</td>
</tr>
<tr>
<td>to identify</td>
<td>38</td>
</tr>
<tr>
<td>to show</td>
<td>37</td>
</tr>
<tr>
<td>to require</td>
<td>36</td>
</tr>
<tr>
<td>to regulate</td>
<td>34</td>
</tr>
<tr>
<td>to suppress</td>
<td>30</td>
</tr>
<tr>
<td>to detect</td>
<td>30</td>
</tr>
<tr>
<td>to express</td>
<td>29</td>
</tr>
<tr>
<td>to activate</td>
<td>24</td>
</tr>
<tr>
<td>to initiate</td>
<td>28</td>
</tr>
<tr>
<td>to encode</td>
<td>20</td>
</tr>
<tr>
<td>to block</td>
<td>18</td>
</tr>
<tr>
<td>to recognize</td>
<td>17</td>
</tr>
<tr>
<td>to stimulate</td>
<td>17</td>
</tr>
<tr>
<td>to increase</td>
<td>7</td>
</tr>
<tr>
<td>to act (as)</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 11: Least common subsumer: the AImedi data set

Using the third feature set containing lemmata from the parent-level provides better results than FS₂. We believe it is due to the fact that in many cases proteins are leaves in a tree so the information about the children is missing.

The best performance on LLL has been obtained by employing the third feature set or the fourth set containing all features as defined in Section 5.2.2. However, we did not find any significant improvements by employing FS₃, FS₄ and FS₅ on the LLL data set (paired t-test, α = 0.05). On AImedi data, moving from one feature set to another (starting with FS₁) always result in significant improvement. Consequently, the best performing method is the one that uses FS₅. In the experiments with
various classification methods in the next section we set the feature set to FS5, unless stated otherwise.

![Graph](image1)

(a) LLL data set

![Graph](image2)

(b) AImed data set

Figure 7: Precision and recall for different feature sets (AdaBoostM1, LLL and AImed data sets)

As we have mentioned above, in most cases the existing approaches to relation extraction provide considerably high precision but low recall. Our results suggest that the approach we have taken leads to higher recall and lower precision. It can be concluded that although the information from the dependency tree levels helps to find many true positives, the local context is sometimes not sufficient to be able to discriminate between true and false positives.
5.4.2 Classifiers

We selected several machine learning algorithms to study how well they perform on the representation that we proposed. In addition, to test the hypothesis that ensemble methods may improve the overall performance, we have conducted experiments with three ensemble methods, stacking, bagging and AdaBoost. A list of diacritics for statistical significance is given in Table 12.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>$0.59 \pm 0.10$</td>
<td>$0.64 \pm 0.11$</td>
<td>$0.61 \pm 0.08$†</td>
</tr>
<tr>
<td>BayesNet</td>
<td>$0.59 \pm 0.09$</td>
<td>$0.70 \pm 0.11$</td>
<td>$0.64 \pm 0.08$†</td>
</tr>
<tr>
<td>IB1</td>
<td>$0.69 \pm 0.12$</td>
<td>$0.67 \pm 0.13$</td>
<td>$0.67 \pm 0.10$</td>
</tr>
<tr>
<td>Stacking</td>
<td>$0.65 \pm 0.12$</td>
<td>$0.69 \pm 0.13$</td>
<td>$0.66 \pm 0.08$</td>
</tr>
<tr>
<td>Bagging</td>
<td>$0.57 \pm 0.09$</td>
<td>$0.68 \pm 0.11$</td>
<td>$0.62 \pm 0.08$†</td>
</tr>
<tr>
<td>AdaBoostM1</td>
<td>$0.70 \pm 0.10$</td>
<td>$0.71 \pm 0.10$</td>
<td>$0.70 \pm 0.08$</td>
</tr>
</tbody>
</table>

† signifiﬁcantly worse than AdaBoostM1
‡ signifiﬁcantly worse than Bagging
§ signifiﬁcantly worse than Stacking
♣ signifiﬁcantly worse than BayesNet

Table 12: A list of diacritics used for statistical signiﬁcance (two-tailed $t$-test, $\alpha = 0.05$)

Table 13: Mean precision, recall, $F_1$ scores and standard deviations on the LLL data set (level-based approach)

We considered three classiﬁers of a different nature, bayesian networks (BayesNet), Naive Bayes and K-nearest neighbor (IB1). Bagging and AdaBoost have been applied to the BayesNet classiﬁer. The experiment with stacking has been constructed in the following way: BayesNet has been chosen as meta-classiﬁer with NaiveBayes and 1-nearest neighbour classiﬁers as individual classiﬁers. The quantitative results of the experiments on both corpora are given in Table 13 and Table 14. In both cases boosting provides much higher recall compared to the supervised methods or other ensemble methods. Bagging helps the least while stacking is placed somewhere between bagging and boosting.

Bunescu and Mooney [16] have also used 10-fold cross-validation to test their methods on the AIMed corpus. They reported on the performance of their approaches by presenting it as a precision-recall curve. To compare our results on the AIMed corpus with the performance of the methods
described in [16], we chose the highest recall received by stacking (0.55 ± 0.05). It corresponds to a precision of around 0.52 on the precision-recall curve which has been received by the subsequence kernel method. We can conclude therefore that our method performs comparably to the subsequence kernel method. The comparison of the results of the best individual classifier (BayesNet) to the subsequence kernel method demonstrates that it performs similarly to it as well. The difference in the performance can be explained by the features used in Bunescu and Mooney [16]. In particular, that approach has considered sequential information which consisted of the words found between two entities, in front of them and after them. After having defined such features, the authors have restricted themselves to subsequences of the types mentioned above, where a maximal word length equals 4. According to Bunescu and Mooney [16], such feature selection leads to less overfitting. In our case, we considered not the common subsequences but the levels from the dependency tree instead. We believe that the selection of levels we have made provides information sufficient for relation learning and constitutes an alternative approach to the method proposed by Bunescu and Mooney [16].

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>0.48 ± 0.04</td>
<td>0.45 ± 0.05</td>
<td>0.46 ± 0.04†‡§♣</td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.47 ± 0.04</td>
<td>0.54 ± 0.05</td>
<td>0.50 ± 0.04†</td>
</tr>
<tr>
<td>IB1</td>
<td>0.49 ± 0.04</td>
<td>0.47 ± 0.05</td>
<td>0.48 ± 0.04§</td>
</tr>
<tr>
<td>Stacking</td>
<td>0.50 ± 0.04</td>
<td>0.56 ± 0.07</td>
<td>0.52 ± 0.04</td>
</tr>
<tr>
<td>Bagging</td>
<td>0.47 ± 0.04</td>
<td>0.53 ± 0.05</td>
<td>0.50 ± 0.04†</td>
</tr>
<tr>
<td>AdaBoostM1</td>
<td>0.52 ± 0.04</td>
<td>0.55 ± 0.05</td>
<td>0.53 ± 0.03</td>
</tr>
</tbody>
</table>

Table 14: Mean precision, recall, F₁ scores and standard deviations on the AImed data set (level-based approach)

Comparison of the results received on the AImed data set with the performance on the LLL data set (Table 13) demonstrates that we have received much better results on the latter. There are several distinctions between the two data sets. Although the LLL data set is much smaller, it contains syntactic information checked by hand. Some classification errors on the LLL data set can be explained by the asymmetry of the relation between genes and proteins.

The feature sets we constructed in our approach are relatively small but able to cover information needed for relation learning. The representation we use results in a training set of the size k * n, where k is the number of features and n is the number of all potential candidates (in our case, it is a Cartesian product of all proteins found within the same sentence multiplied by the number of sentences). The largest feature set we employ
86 feature engineering using syntactic structures

consists of 9 features, thus the maximal size of a training set equals $9 \times n$. Therefore, the training time depends mostly on the machine learning method used. It is, for example, known that k-nn algorithm is slower than Naive Bayes, although both of them can be applied to the representation we propose.

5.4.3 A note on syntactic analysis

As we already mentioned, approaches making use of syntactic structure depend on the accuracy of the parser. Precision and recall of the parsers we used varies and it is likely that some errors in classification are due to incorrect parsing. To explore if it affects the accuracy of found interactions, we conducted the following experiments. In addition to the LinkParser (whose output was checked by hand), we used the Minipar and Charniak parser to analyze the LLL data set. We fixed the feature set (FS4) and used the same machine learning method (AdaBoostM1) on the representation received by all three parsers.

![Comparison among LinkParser, Minipar and Charniak](image)

Figure 8: Precision and recall for LinkParser, Minipar and the Charniak parser (AdaBoostM1, the LLL data set)

As expected, use of the LinkParser output provides the highest precision and recall for relation learning on the LLL data set (Fig. 8). There is consequently a statistically significant difference between the run that uses the LinkParser and any other parser (two-tailed t-test, $\alpha = 0.05$). Interestingly, no matter what parser is employed, there is only a little difference between precision and recall.
We already mentioned in Section 5.3.3 that all three parsers give different output. It can be demonstrated on the example in Appendix I of this chapter (Table 15 and Table 16). In particular, Minipar produces syntactic functions for each dependency, whereas the modified output of the Charniak parser lacks them. To test whether syntactic functions contribute to the learning task, we conducted another experiment using the same feature set (FS4) and the AdaBoost classifier with syntactic functions and without them. To make such comparison fair, we considered the LinkParser only. The output of Minipar was not checked by hand so it is not possible to analyze whether the changes in performance are due to the incorrect parsing or to the presence (absence) of the syntactic functions. The results are presented in Fig. 9. Removing syntactic functions decreases both recall and precision and these changes are significant (two-tailed t-test, $\alpha = 0.05$).

Figure 9: Impact of syntactic functions on the level-based approach (LinkParser, the LLL data set)

In addition, all three parsers use different sets of syntactic functions. Sometimes, syntactic functions are grouped or discriminated according to a certain criterion. For instance, such phrases as ‘transcribed by polymerase’ and ‘activation in prespore’ are treated differently by Minipar and LinkParser. While the former introduces a special link $pcomp-n$ between a preposition $in$ and a noun $prespore$, the latter incorporates preposition in the syntactic function and outputs the relation $comp-in$ between $activation$ and $prespore$. Such treatment of complements means that the actual training sets differ. LinkParser seems to provide more useful information, because a parent node of $prespore$ is $activation$ and not $in$ as in case of the Minipar or the Charniak parser.
The performance of current state-of-the-art parsers on biomedical data has been studied by Grover et al. [63]. The impact of currently available parsing technology on relation extraction has also been investigated by Rinaldi et al. [152]. In this work a broad-coverage probabilistic parser LT Chunk is used to obtain verbal and nominal chunks. Interestingly, the results that Rinaldi et al. [152] have achieved with correct syntactic analysis do not differ much from the results that relied on automatic syntactic analysis. In our case we can clearly see a difference between performance that was attained by using correct analysis and the results which were obtained by automatic analysis. Nevertheless, the parsers that we studied here differ in their output and it may be the case that some differences in the final performance can be attributed to this factor.

5.5 CONCLUSIONS

In this chapter, we have proposed a novel representation for learning relations based on dependency trees. Learning relations differs from extracting named entities since it is not easy to represent a relation instance in the commonly used attribute-value format. This, in turn, makes it impossible to employ many well-known machine learning methods. To overcome this problem, either pattern-based methods or methods capable of using complex representations, such as support vector machines [208] have been proposed. The representation we use is derived from the complex structures (dependency trees) but it is still an attribute-value like representation.

Another advantage this representation gives us lies in the possibility to use ensemble methods. Ensemble methods are methods whose main purpose is to combine the decisions of individual classifiers. When conducting experiments using bagging, stacking and boosting, the performance improved, mainly contributing to higher recall. It also motivates for combining such methods with approaches providing high precision.

We have tested the representation on the data sets containing interactions between genes and proteins (LLL data set) and between proteins (AIMed data set). The results on both data sets are promising and comparable to the state-of-art results.

Our experiments suggest that the quality of syntactic analysis is of vital importance when using techniques based on it. As it has been shown, the syntactic analysis verified by experts is more accurate and this, in turn, leads to higher accuracy of the extracted interactions.
(5.8) ykuD was transcribed by SigK RNA polymerase from T4 of sporulation.

Table 15: Syntactic functions used by LinkParser for Example (5.8)
90 feature engineering using syntactic structures

transcribed
  s  be  by_subj
  ykuD
  was  by
  pcomp-n
  polymerase
  nn  mod
  RNA  from
  lex-mod  pcomp-n
  SigK  T4
  mod
  of
  pcomp-n
  sporulation

Figure 11: Minipar’s output for Example 5.8

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>surface subject</td>
<td>(transcribed, ykuD)</td>
</tr>
<tr>
<td>be</td>
<td>be</td>
<td>(transcribed, was)</td>
</tr>
<tr>
<td>by-subj</td>
<td>by-subj (for passive voice)</td>
<td>(transcribed, by)</td>
</tr>
<tr>
<td>pcomp-n</td>
<td>nominal complement of prepositions</td>
<td>(by, polymerase), (from, T4), (of, sporulation)</td>
</tr>
<tr>
<td>nn</td>
<td>complement</td>
<td>(polymerase, RNA)</td>
</tr>
<tr>
<td>mod</td>
<td>modifier</td>
<td>(polymerase, from)</td>
</tr>
<tr>
<td>lex-mod</td>
<td>modifier</td>
<td>(RNA, SigK)</td>
</tr>
</tbody>
</table>

Table 16: Syntactic functions used by MiniPar for Example 5.8
was

$ykuD$ transcribed by polymerase RNA from $\text{SigK}$ of $T_4$ of sporulation

Figure 12: Charniak parser’s output for Example (5.8)

<table>
<thead>
<tr>
<th>Verb</th>
<th>Number of Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>to activate</td>
<td>5</td>
</tr>
<tr>
<td>to regulate</td>
<td>4</td>
</tr>
<tr>
<td>to induce</td>
<td>3</td>
</tr>
<tr>
<td>to depend</td>
<td>2</td>
</tr>
<tr>
<td>to act</td>
<td>2</td>
</tr>
<tr>
<td>to require</td>
<td>2</td>
</tr>
<tr>
<td>to repress</td>
<td>2</td>
</tr>
<tr>
<td>to recognize</td>
<td>2</td>
</tr>
<tr>
<td>to inhibit</td>
<td>2</td>
</tr>
<tr>
<td>to derive</td>
<td>2</td>
</tr>
<tr>
<td>to bind</td>
<td>2</td>
</tr>
<tr>
<td>to terminate</td>
<td>2</td>
</tr>
<tr>
<td>to initiate</td>
<td>2</td>
</tr>
<tr>
<td>to increase</td>
<td>2</td>
</tr>
<tr>
<td>to govern</td>
<td>2</td>
</tr>
<tr>
<td>to encode</td>
<td>2</td>
</tr>
<tr>
<td>to contribute</td>
<td>2</td>
</tr>
<tr>
<td>to convert</td>
<td>2</td>
</tr>
<tr>
<td>to block</td>
<td>2</td>
</tr>
<tr>
<td>to affect</td>
<td>2</td>
</tr>
<tr>
<td>to couple</td>
<td>2</td>
</tr>
<tr>
<td>to reduce</td>
<td>2</td>
</tr>
<tr>
<td>to limit</td>
<td>2</td>
</tr>
<tr>
<td>to prevent</td>
<td>2</td>
</tr>
<tr>
<td>to dephosphorylate</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 13: Distribution of the verbs selected by experts from the LLL data set