Not all links are equal: Exploiting Dependency Types for the Extraction of Protein-Protein Interactions from Text

Philippe Thomas [1], **Stefan Pietschmann** [1], **Illés Solt** [2], **Domonkos Tikk** [1,2] **and Ulf Leser** [1]
[1]Knowledge Management in Bioinformatics, Institute for Computer Science,
Humboldt-University of Berlin,
Unter den Linden 6, 10099 Berlin, Germany
[2]Department of Telecommunications and Media Informatics,
Budapest University of Technology and Economics,
Magyar tudósok körútja 2, 1117 Budapest, Hungary
{thomas,pietschm,solt,tikk,leser}@informatik.hu-berlin.de

Abstract

The extraction of protein-protein interactions (PPIs) reported in scientific publications is one of the most studied topics in Text Mining in the Life Sciences, as such algorithms can substantially decrease the effort for databases curators. The currently best methods for this task are based on analyzing the dependency tree (DT) representation of sentences. Many approaches exploit only topological features and thus do not yet fully exploit the information contained in DTs. We show that incorporating the grammatical information encoded in the types of the dependencies in DTs noticeably improves extraction performance by using a pattern matching approach. We automatically infer a large set of linguistic patterns using only information about interacting proteins. Patterns are then refined based on shallow linguistic features and the semantics of dependency types. Together, these lead to a total improvement of 17.2 percent points in F_1, as evaluated on five publicly available PPI corpora. More than half of that improvement is gained by properly handling dependency types. Our method provides a general framework for building task-specific relationship extraction methods that do not require annotated training data. Furthermore, our observations offer methods to improve upon relation extraction approaches.

1 Introduction

Insights about protein-protein interactions (PPIs) are vital to understand the biological processes within organisms. Accordingly, several databases, such as IntAct, DIP, or MINT, contain detailed information about PPIs. This information is often manually harvested from peer reviewed publications (Ceol et al., 2010). However, it is assumed that a high amount of PPIs is still hidden in publications. Therefore, the automated extraction of PPIs from text has attracted considerable attention from biology research.

A number of different techniques have been proposed to solve the problem of extracting PPIs from natural language text. These can be roughly organized into one of three classes: co-occurrence, machine learning, and pattern matching (for a recent survey, see (Zhou and He, 2008)). The co-occurrence based approaches use only information on the co-existence of protein mentions in a given scope. They are easy to implement and allow for efficient processing of huge amounts of texts, but they are also prone to generate many false positives because they cannot distinguish positive from negative pairs. The second class is based on machine learning. Here, a statistical model is learned from a set of positive and negative examples and then applied to unseen texts. In general, machine learning-based methods to relation extraction perform very well for any task where sufficient, representative and high quality training data is available (Kazama et al., 2002). This need for training data is their major drawback, as annotated texts are, especially in the Life Sciences, rather costly to produce. Furthermore, they are prone to over-fit to the training corpus, which renders evaluation results less inferable to real applications. A third class of methods is based on pattern matching. Such methods work with patterns constructed from linguistically anno-

Proceedings of the 2011 Workshop on Biomedical Natural Language Processing, ACL-HLT 2011, pages 1–9,
Portland, Oregon, USA, June 23-24, 2011. ©2011 Association for Computational Linguistics

tated text, which are matched against unseen text to detect relationships. Patterns can either be inferred from examples (Hakenberg et al., 2010; Liu et al., 2010) or can be defined manually (Fundel et al., 2007). Systems based on manually defined patterns typically use few patterns, leading to high precision but low recall (Blaschke et al., 2002). In contrast, systems that learn patterns automatically often produce more patterns and exhibit a better recall, at the cost of a decrease in precision. To circumvent this penalty, several works have tried to improve patterns. E.g., SPIES (Hao et al., 2005) filters patterns using the minimum description length (MDL) method which improves its F_1 by 6.72%.

Another classification of PPI extraction methods is based on the sentence representation that is applied. The simplest such representation is the bag of words (BoW) that occur in the sentence; more complex representations are constituent trees, capturing the syntactic structure of the sentence, and dependency trees (DTs), which represent the main grammatical entities and their relationships to each other. PPI extraction methods use various sentence representation, e.g., are based only on BoW (Bunescu and Mooney, 2006; Giuliano et al., 2006), use only DTs (Erkan et al., 2007), or combine representations (Airola et al., 2008; Miwa et al., 2008).

In the last years, dependency trees have become the most popular representation for relation extraction. DTs characterize, via their dependency links, grammatical relationships among words. They are particularly favored by kernel-based learning approaches, see e.g. (Culotta and Sorensen, 2004; Erkan et al., 2007; Airola et al., 2008; Miwa et al., 2008; Kim et al., 2010) but also graph matching approaches using DTs have been proposed (Liu et al., 2010). However, these methods do not further utilize the grammatical information encoded in the dependency types (edge labels). Recently proposed methods like (Buyko et al., 2009; Rinaldi et al., 2010) modify the DTs by e.g. trimming irrelevant dependencies. In contrast to these approaches, our method *exploits* the dependency types of DTs and performs basic transformations on DTs; we use Stanford dependencies, which are presumably the most often used DT representation in PPI extraction.

The rest of this paper is organized as follows. We describe our novel method for extracting PPIs from text that is based on pattern matching in dependency graphs. We evaluate our method against benchmark PPI corpora, and discuss results with a focus on dependency type information based methods.

2 Methods

Our approach consists of a series of steps: First, we extract sentences from Medline and PMC open access that contain pairs of genes/proteins known to interact. Second, we convert each of those sentences into DTs and derive putative tree patterns for each pair. Having a set of such patterns, we apply a number of generalization methods to improve recall and filtering methods to improve precision. We discern between methods that are purely heuristic (termed shallow) and steps that incorporate dependency type information (termed grammatical). To predict PPIs in unseen text, the resulting patterns are matched against the corresponding DTs.

2.1 Extraction of PPI sentences

We apply the method described in (Hakenberg et al., 2006) to extract a set of sentences from Medline and PMC potentially describing protein interactions. Essentially, this method takes a database of PPIs (here IntAct; (Aranda et al., 2010)) and searches all sentences in Medline and PMC containing any of those pairs. Proteins were tagged and normalized using GNAT (Hakenberg et al., 2008). To avoid a possible bias, articles contained in any of the five evaluation corpora are excluded. This resulted in 763,027 interacting protein pairs.

2.2 Pattern generation and matching

For each protein pair we generate a new sentence and apply entity blinding, meaning that named entities are replaced by placeholders to avoid systemic bias. Specifically, the mentions of the two proteins known to interact are replaced by the placeholder *ENTITY_A* and any additional proteins in the same sentence are replaced by *ENTITY_B*. Tokens are tagged with their part-of-speech (POS) using MedPost (Smith et al., 2004), which is specifically optimized for biomedical articles. Constituent parse trees are generated using the Bikel parser (Bikel, 2002) and converted to DTs by the Stanford converter (De Marneffe et al., 2006). In a DT, the shortest path between two tokens is often assumed to contain the most valuable information about their mutual relationship. Therefore, we generate a pattern from each DT by extracting the shortest, undirected path between the two occurrences of *ENTITY_A*. The set of initial patterns is denoted by S_{IP}.

We employ several methods to improve the quality of this initial set of patterns. We systematically evaluated possible constellations and identified those that help in improving performance of PPI extraction. The modifications are of two kinds. Pattern generalizers are intended to elevate recall, whereas pattern filters should raise precision. We present two types of methods: Shallow methods are simple heuristics whereas grammatical methods are rules that exploit the information in dependency types.

We use a strict graph matching approach for pattern matching. We consider a pattern to match a subgraph of a DT iff all their nodes and edges match exactly, including edge labels and edge directions.

2.3 Pattern generalization

It is a common practice in NLP to apply some preprocessing on patterns to reduce corpus-specificity. In particular, we perform stemming (G_{ST}), removal of common protein name prefixes and suffixes (G_{PN}), and replacement of interaction phrases by single words (G_{IW}). We summarize these steps as *shallow generalization* steps. We only describe the latter two (G_{PN}, G_{IW}) in more detail.

Protein names are often modified by suffixes like *-kinase*, *-receptor* or *-HeLa* or by prefixes like *phospho-* or *anti-*. These affixes are usually not covered by entity blinding as the entity recognition method does not consider them as part of the protein name. As such affixes are not relevant for relation extraction but do interfere with our exact graph matching approach, we apply the G_{PN} heuristic to remove them.

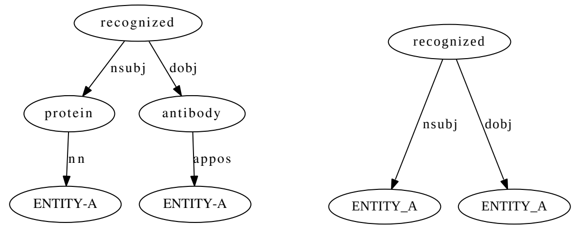
Interactions between proteins can be expressed very diversely in natural language. In almost all cases there is at least one word that specifies the interaction semantically, called the *interaction word*; this is often a verb, such as "binds" or "phosphorylates", but can as well be a noun, such as "[induced] phosphorylation", or an adjective, such as "binding". The G_{IW} heuristic generalizes patterns by substituting all contained interaction words with generic placeholders. We assembled a list of 851 interaction words (including inflection variants) based on (Temkin and Gilder, 2003; Hakenberg et al., 2006) that was further enriched manually. Based on their POS-tags, interaction words are assigned to one of the three placeholders *IVERB*, *INOUN*, *IADJECTIVE*. We also experimented with a single interaction word placeholder, *IWORD*, handling the case of incorrect POS-tags.

Unifying dependency types (G_{UD}): The Stanford typed dependency format contains 55 grammatical relations organized in a generalization hierarchy. Therefore, it is a natural idea to treat similar (e.g., sibling) dependency types equally by replacing them with their common parent type. We manually evaluated all dependency types to assess whether such replacements are viable. The final list of replacements is listed in Table 1. Note that we used the so-called collapsed representation of dependency types of the Stanford scheme. This means that prepositional and conjunctive dependencies are collapsed to form a single direct dependency between content words, and the type of this dependency is suffixed with the removed word. In the G_{UD} generalizer, these dependency subtypes are substituted by their ancestors (e.g., `prep`, `conj`).

Dependency types	Common type
`subj`, `nsubj*`, `csubj*`	`subj`
`obj`, `dobj`, `iobj`, `pobj`	`obj`
`prep_*`, `agent`, `prepc`	`prep`
`nn`, `appos`	`nn`

Table 1: Unification of specific dependency types to a single common type by the generalizer G_{UD}. Note that `agent` is merged with dependency type `prep` as it is inferred for the preposition "by".

Collapsing dependency links (G_{CD}): In addition to the collapsing performed by Stanford converter, we remove edges that likely are irrelevant for PPIs. We focused on removing the dependencies `nn` (noun compound modifier) and `appos` (appositional modifier). These grammatical constructions have the same syntactic role but they carry somewhat different meaning. They function as noun phrase modifiers and often specify the subtype of an entity, which is irrelevant for our task. As these two dependency types convey no information about the interaction itself, the dependency itself and the corresponding noun can be removed, as long as the noun is not an entity itself. As an example, this generalizer is applied on the dependency parse tree of the sentence "*ENTITY_A* protein recognized antibody (*ENTITY_A*)" shown on Figure 1a. The modified parse tree is depicted on Figure 1b.



(a) Original pattern (b) Generalized pattern

Figure 1: Dependency pattern before and after collapsing `nn` and `appos` dependency links using the generalizer G_{CD}.

2.4 Pattern constraints

Due to the automatic construction method, our set of patterns also contains samples derived from sentences that do not actually describe an interaction between proteins, although it does contain a pair of interacting proteins. Such patterns lead to wrong matches. As a countermeasure, we define constraints an extracted pattern has to fulfill. Patterns not adhering to these constraints are removed from the pattern set, thus increasing precision. Standard (shallow) heuristics for doing so are the exclusion of negation words (C_{NW}) and the restriction to patterns containing interaction-related words from a predefined set (C_{IW}). Patterns containing negations potentially match two negative protein pairs. Such pattern can be removed to prevent wrong extractions. For negation words, the list described in (Fundel et al., 2007) was used. Additionally, patterns containing the dependency type *conj_no**, *conj_or*, or *prep_without* are also removed. On top of those previously known approaches, we developed two new filter to leverage the semantic richness of the DTs.

Dependency combination (C_{DC}): Interaction words are organized into the following categories: *verb*, *adjective* and *noun*. Based on linguistic considerations we define "dependency patterns" for the different word types. For example we assume that interaction verbs describe an action that originates in one protein and affects the other protein. Obviously, the dependency combination `subj` with `obj` fulfills this consideration (for an example see Figure 1b). We manually evaluated a few DTs containing PPI for each interaction word category (verb, noun, adjective) and determined all combinations of dependency types that are valid for the given category. The resulting combinations are listed in Table 2.

Part of speech	Dependency type combination	
Noun	prep	prep
	prep	nn
	prep	amod
	nn	nn
	nn	amod
Verb	prep	subj
	prep	infmod
	prep	partmod
	obj	subj
	obj	infmod
	obj	partmod
Adjective	amod	

Table 2: Allowed dependency type combinations based on classes of POS classes (constraint C_{DC}). `subj` = {`nsubj`, `nsubjpass`, `xsubj`, `csubj`, `csubjpass`}, `obj` = {`dobj`, `pobj`, `iobj`} and `prep` = {`prep_*`, `agent`}

Syntax Filter (C_{SF}): A particular case in PPI extraction are sentences with enumerations, as shown in Figure 2. Such (possibly quite long; the longest enumeration we found contains not less than 9 proteins) enumerations greatly increase the number of protein pairs.

We observed that sentences in which the common dependency type is `prep_between` or `nn` often do describe an association between the connected proteins. Accordingly, such patterns are retained.

The remaining pairs inside enumerations often do not describe an interaction between each other. Therefore, we developed a special handling of enumerations, based on dependency types. If two proteins have a common ancestor node connected by the same dependency type, we assume that those proteins do not interact with each other. Accordingly, we remove all such patterns.

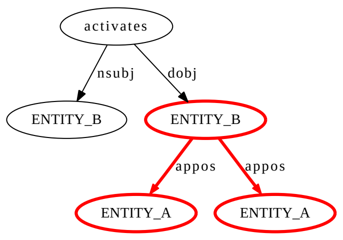


Figure 2: Dependency tree (DT) for the entity blinded sentence "*ENTITY_B* activates *ENTITY_B*, *ENTITY_A*, *ENTITY_A*." with the initial pattern highlighted in bold red. Application of C_{SF} removes this pattern.

3 Results

For evaluation we use five manually annotated benchmark corpora: AIMed, BioInfer, HPRD50, IEPA, and LLL. Those have been unified to the "greatest common factors" by Pyysalo et al. (2008). This means that protein names in the corpora are tagged and that interactions are undirected and binary. A basic overview of the corpora can be found in Table 1 of (Airola et al., 2008).

A sentence with n entities contains $\binom{n}{2}$ different undirected entity pairs. For each entity pair in a sentence, we generate a separate evaluation example, apply entity blinding and generate the DT in the same manner as previously described for generating the pattern set. All patterns are then matched against the DTs of the sentences from the evaluation corpora. If at least one pattern matches, the pair is counted as *positive* otherwise as *negative*. From this information we calculate precision, recall, and F_1 scores.

Table 3 shows results using the initial pattern set and the different configurations of generalized / filtered pattern sets. We evaluate the impact of shallow and grammar-based methods separately. Recall that $S_{shallow}$ encompasses stemming (G_{ST}), substitution of interaction words (G_{IW}), suffix/prefix removal at entity names (G_{PN}), and interaction (C_{IW}) and negation word filtering (C_{NW}), while $S_{grammar\text{-}based}$ encompasses unification of dependency types (G_{UD}), collapsing dependency links (G_{CD}), the dependency combination constraint (C_{DC}) and the syntax filter (C_{SF}). In addition, results after application of all generalizers $S_{generalizers}$, all constraints $S_{constraints}$

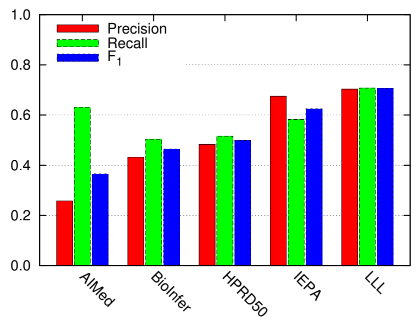


Figure 3: Results for the five corpora using the setting achieving highest overall F_1 (S_{all}).

and the combination of both S_{all} are also included. Corpus-specific results for the best setting in terms of F_1 (S_{all}) are shown in Figure 3.

Setting		P	R	F_1	#
Baseline	$S_{initial}$	23.2	34.8	27.8	478 k
Generalizers	G_{PN}	23.4	37.0	28.7	423 k
	G_{IW}	26.2	45.3	33.2	453 k
	G_{ST}	24.1	37.4	29.3	471 k
	G_{UD}	24.0	38.3	29.5	467 k
	G_{CD}	26.3	48.9	34.2	418 k
Constraints	C_{NW}	23.4	34.8	28.0	455 k
	C_{IW}	42.5	17.2	24.5	322 k
	C_{DC}	39.5	15.9	22.7	318 k
	C_{SF}	28.2	32.6	30.3	419 k
Combinations	$S_{generalizers}$	28.2	69.0	39.9	290 k
	$S_{constraints}$	68.3	12.7	21.4	224 k
	$S_{shallow}$	40.9	31.4	35.5	253 k
	$S_{grammar\text{-}based}$	33.2	42.1	37.2	264 k
	S_{all}	38.2	54.8	45.0	152 k

Table 3: Performance of pattern sets on the ensemble of all five corpora. # denotes the pattern set size.

4 Discussion

We presented a pattern-based approach to extract protein-protein interactions from text. Our main contribution in this paper was a systematic study on the usage of dependency types within this approach. We showed that using such knowledge, F_1 on average improves by 9.4 percentage points (pp) (27.8 % to 37.2 %) as measured on the five benchmark PPI corpora.

Apart from this result, we note that our method also has a number advantageous features: First, patterns are learned from co-mentions of pairs of proteins known to interact, and hence no annotated corpus is necessary. This does not only make an application of the method for new tasks easier and cheaper, but also prevents over-fitting to a training corpus. Note, that as shown recently by (Airola et al., 2008; Tikk et al., 2010), essentially all state-of-the-art machine learning methods show large performance differences depending on whether or not the evaluation and training examples are drawn from the same corpus. In particular, cross-validation results of those are consistently more optimistic than the more realistic cross-learning results. In contrast, a pattern-based approach like ours is not prone to over-fitting. Furthermore, debugging our method is rather simple. Unlike when using a black-box machine learning method, whenever a false positive match is found, one can pinpoint the specific pattern producing it and take action.

The work most closely related to ours is RelEx (Fundel et al., 2007). RelEx uses a small set of fixed rules to extract directed PPIs from dependency trees. Some of these rules also take advantage of dependency types, for instance, to properly treat enumerations. A reimplementation of RelEx (Pyysalo et al., 2008) was recently evaluated on the same corpora we used (see Table 7) and was found to be on par with other systems, though some of its measures were considerably worse than those reported in the original publication. Compared to our approach, RelEx is similar in that it performs pattern matching on DTs using information encoded in dependency types, however, there are some notable differences: First, RelEx rules were defined manually and are highly specific to protein-protein interactions. It is not clear how these could be adapted to other applications; in contrast, we described a general method that performs pattern learning from automatically generated examples. Second, it is not clear how RelEx results could be further improved except by trial-and-error with more rules. In contrast, our pattern learning method offers a natural way of improvement by simply increasing the number of examples and hence the number of patterns. We compared the results of our approach using an increasingly larger pattern set and observed a continuous increase in F_1, due to a continuous improvement in recall. Consequently, using more PPI databases would likely produce better results. Third, our generalization methods can be seen as graph rewriting rules. The result of applying them to a DT is, again, a DT; thus, they can easily be used as pre-processing coupled with other PPI extraction methods (a direction we are currently exploring). In contrast, RelEx matches patterns against DTs, but cannot be used to transform DTs.

In the following, we discuss the impact of the refinement methods individually and provide a brief error analysis based on a random sample of false negative pairs and a random sample of low precision patterns. We also compare our best results with those of several state-of-the-art methods.

4.1 Generalizers and constraints

As can be seen in Table 3, all of the generalizers increased recall and even provide minor improvement in precision. For the combination of all five generalizers ($S_{\text{generalizers}}$), an overall increase of 34.2 pp in recall and 5 pp in precision was observed. From the shallow generalizers, merging interaction phrases (G_{IW}) was proven to be the most effective, accounting for an increase of 10.5 pp in recall and 3 pp in precision. As shown in Table 4, the general variant, which merges all interaction phrases to a common word, is slightly superior to the variant in which interaction words are merged class-wise.

G_{IW} variant	P	R	F_1
Specific	26.1	44.7	33.0
General	26.2	45.4	33.2

Table 4: Results for collapsing interaction word variants (G_{IW}).

For the grammar-based generalizer unifying dependency types (G_{UD}), each of the different variants was evaluated separately (see Table 5). The combination of the all different variants leads to an increase of 3.5 pp in recall. As shown in Table 6, collapsing the dependency types `nn` and `appos` (G_{CD}) also provides the highest improvement when applied in combination.

In contrast to generalizers that alter patterns, constraints remove patterns from the pattern set. As shown in Table 3, application of all constraints

G_{UD} variant	P	R	F_1
`subj`	23.4	35.1	28.1
`obj`	23.3	34.9	27.9
`prep`	24.0	37.0	29.1
`nn`	23.1	35.6	28.1
`sopn`	24.0	38.3	29.5

Table 5: Dependency type aggregations used in generalizer G_{UD}. `sopn` combines the dependency aggregations for `subj`, `obj`, `prep`, and `nn`.

G_{CD} variant	P	R	F_1
`appos`	23.6	38.1	29.2
`nn`	25.8	45.3	32.9
`appos+nn`	26.3	48.9	34.2

Table 6: Impact of collapsing the dependency types `appos` and `nn` using generalizer G_{CD}.

($S_{constraints}$) leads to an increase in precision of 45.1 pp at the cost of 22.1 pp reduced recall.

The shallow constraint that disallows patterns with negation words (C_{NW}) has comparably little impact and removes only 5 % of the patterns. In contrast, the interaction word constraint (C_{IW}) is less conservative and removes more than 32.6 % of the patterns, trading off an increase of 19.3 pp in precision to a decrease of 17.6 pp in recall.

Among the grammar-based constraints, the dependency combination constraint C_{DC} is superseded by the syntax filter constraint (C_{SF}) that removes 12 % of the patterns and increases precision about 5 pp while recall drops by only 2.2 pp. Note that C_{SF} is the only constraint substantially increasing F_1.

As seen in Table 3, combinations of generalizers and constraints yield almost fully additive improvements. The combination of all shallow refinements only ($S_{shallow}$) leads to an increase of 7.7 pp in F_1, whereas with the addition of our grammar-based refinements (S_{all}) a total increase of 17.2 pp in F_1 is achieved. We justify that the inclusion of dependency type information adds a source of knowledge that further improves on conventional methods such as stemming or negation filtering.

4.2 Comparison with other methods

We compare the results of our best setting (S_{all}) with the results of the recently published analysis of nine state-of-the-art machine learning methods for PPI extraction (Tikk et al., 2010). For these methods, a cross-learning evaluation by training each kernel on the ensemble of four corpora and evaluating it on the fifth has been performed. Detailed results are given in Table 7. In terms of F_1 we achieve on BioInfer, the corpus with most protein pairs, the second-best result. On IEPA and LLL we achieve mid-range results and on AIMed and HPRD50 we yield results below average. Overfitting remains a severe problem in ML based methods as these results are inferior to those measured in cross-validation (Tikk et al., 2010), though there are suggestions to overcome this issue even in a ML setting (Miwa et al., 2009).

4.3 Error analysis

We randomly picked 30 gold standard sentences (all corpora) containing false negatives pairs and investigated all 72 false negative pairs included therein. For 29 positive pairs, possibly matching pattern were removed by C_{DC}, as the corresponding dependency combination was not covered in our rule set. Further 16 graphs passed the filtering, but our set of sentences contained no adequate pattern. The third largest fraction of errors (13 cases) are pairs which, by our understanding, hardly describe an interaction. In 11 cases, the dependency parse trees are incorrect and therefore they do not provide the correct syntactic information. For 7 pairs, the shortest path covers insufficient syntactic information to decide whether two proteins interact. For instance Figure 4 provides not enough information on the shortest path, whereas the second shortest path would provide sufficient information. Finally, three pairs were filtered by the C_{IW} filter, as their interaction words were missing from our list.

We conclude that some constraints (especially C_{DC} and C_{IW}) are too aggressive. Relaxation of these syntactic rules should lead to higher recall.

We also analyzed the 30 patterns producing the most false positives matches. 20 of them contained an interaction verb, the remaining 10 an interaction noun. The 10 noun patterns produced more than twice as many false positives as the 20 verb patterns while matching about 50 % less true positives. The single noun pattern producing the most false positives (356) can be seen on Figure 5a. Among the 10, four additional patterns can be seen as an extension

Method	AIMed			BioInfer			HPRD50			IEPA			LLL		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Shallow linguistic (Giuliano et al., 2006)	28.3	**86.6**	42.6	**62.8**	36.5	46.2	56.9	68.7	62.2	71.0	52.5	60.4	79.0	57.3	66.4
Spectrum tree (Kuboyama et al., 2007)	20.3	48.4	28.6	38.9	48.0	43.0	44.7	**77.3**	56.6	41.6	9.6	15.5	48.2	**83.5**	61.2
k-band shortest path (Tikk et al., 2010)	28.6	68.0	40.3	62.2	38.5	**47.6**	61.7	74.2	67.4	72.8	**68.7**	**70.7**	83.7	75.0	**79.1**
Cosine distance (Erkan et al., 2007)	27.5	59.1	37.6	42.1	32.2	36.5	63.0	56.4	59.6	46.3	31.6	37.6	80.3	37.2	50.8
Edit distance (Erkan et al., 2007)	26.8	59.7	37.0	53.0	22.7	31.7	58.1	55.2	56.6	58.1	45.1	50.8	68.1	48.2	56.4
All-paths graph (Airola et al., 2008)	30.5	77.5	43.8	58.1	29.4	39.1	64.2	76.1	**69.7**	**78.5**	48.1	59.6	**86.4**	62.2	72.3
RelEx reimpl. (Pyysalo et al., 2008)	**40.0**	50.0	**44.0**	39.0	45.0	41.0	**76.0**	64.0	69.0	74.0	61.0	67.0	82.0	72.0	77.0
Our method (S_{all})	25.8	62.9	36.6	43.4	**50.3**	46.6	48.3	51.5	49.9	67.5	58.2	62.5	70.3	70.7	70.5

Table 7: Cross-learning results. Classifiers are trained on the ensemble of four corpora and tested on the fifth one (except for the rule-based RelEx). Best results are typeset in bold.

of this pattern leading to a total amount of 732 false positives while only 172 true positives. This phenomenon is caused by the way in which generalizers and constraints are currently applied. The unification of different `prep_*` dependency types to the general `prep` (G_{UD}) makes some dependency type combinations indistinguishable, e.g. (`prep_to`, `prep_to`) and (`prep_to`, `prep_of`). The dependency type combination constraint (C_{DC}) would disallow a pattern containing the first combination, but as it is not applied in the matching phase, its benefits cannot be realized. A lesson learned from this example is that constraints should also be applied in the matching step as follows. After a successful match, the constraints should be applied to the original ungeneralized counterparts of the matching subgraphs. Similar conclusions can be drawn from examining the verb pattern producing the most false positives shown in Figure 5b.

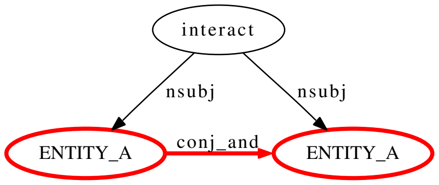
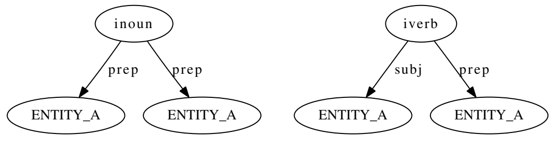


Figure 4: Example dependency parse where the information extracted by the shortest path (highlighted in bold red) is insufficient.



(a) Noun pattern (b) Verb pattern

Figure 5: Patterns producing the most false positives. Depicted dependency types are generalized according to G_{UD} and G_{IW}.

5 Conclusion

We presented a pattern-based approach to extract PPIs from literature. Its unique features are the capability of learning patterns from ”cheap” training data, i.e., co-mentions of proteins known to interact, and the use of linguistically motivated refinements on the dependency structures of the DT it operates on. We utilized the fact that not all dependency types are of equal importance for relation extraction; for instance, collapsing dependency links (G_{CD}) or unifying dependencies (G_{UD}) considerably improved extraction performance. However, as our error analysis shows, especially our filtering methods deserve further study. Even without manually annotated training data, our approach performs on par with state-of-the-art machine learning methods when evaluated in a cross-learning setting. In particular, it reaches the second best performance (in terms of F-measure) of all approaches on the largest PPI corpus.

Apart from presenting a volatile and elegant new method for relationship extraction, we believe that our study on using dependency type information also will be helpful for advancing the performance of other methods. For instance, we are currently experimenting with using our DT-rewrite rules as a preprocessor for a kernel-based extraction method.

Acknowledgments

PT was supported by BMBF, grant No 0315417B; IS was supported by DAAD; DT was supported by Alexander von Humboldt-Foundation.

References

A. Airola, S. Pyysalo, F Ginter J. Björne, and T. Pahikkala. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics*, 9:S2.

B. Aranda, P. Achuthan, Y. Alam-Faruque, I. Armean, A. Bridge, C. Derow, M. Feuermann, et al. 2010. The IntAct molecular interaction database in 2010. *Nucleic Acids Res*, 38:D525–D531, Jan.

DM. Bikel. 2002. Design of a Multi-lingual, Parallel-processing Statistical Parsing Engine. In *In Human Language Technology Conference*, pages 24–27.

C. Blaschke, L. Hirschman, and A. Valencia. 2002. Information extraction in molecular biology. *Briefings in Bioinformatics*, 3(2):154–165.

R. Bunescu and R. Mooney. 2006. Subsequence Kernels for Relation Extraction. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 171–178. MIT Press, Cambridge, MA.

E. Buyko, E. Faessler, J. Wermter, and U. Hahn. 2009. Event extraction from trimmed dependency graphs. In *BioNLP'09*, pages 19–27.

A. Ceol, Chatr AA., L. Licata, D. Peluso, L. Briganti, L. Perfetto, L. Castagnoli, and G. Cesareni. 2010. MINT, the molecular interaction database: 2009 update. *Nucl. Acids Res.*, 38(suppl1):D532–539.

A. Culotta and JS. Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *ACL'04*, pages 423–429.

MC. De Marneffe, B.Maccartney, and CD. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *In LREC 2006*.

G. Erkan, A. Özgür, and DR. Radev. 2007. Semi-Supervised Classification for Extracting Protein Interaction Sentences using Dependency Parsing. In *Proc. of EMNLP–CoNLL'07*, pages 228–237.

K. Fundel, R. Küffner, and R. Zimmer. 2007. RelEx – relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, February.

A. Giuliano, A. Lavelli, and L. Romano. 2006. Exploiting Shallow Linguistic Information for Relation Extraction from Biomedical Literature. In *Proc. of EACL'06*, pages 401–408, Trento, Italy. The Association for Computer Linguistics.

J. Hakenberg, U. Leser, H. Kirsch, and D. Rebholz-Schuhmann. 2006. Collecting a large corpus from all of Medline. In *SMBM'06*, pages 89–92, April.

J. Hakenberg, C. Plake, R. Leaman, M. Schroeder, and G. Gonzalez. 2008. Inter-species normalization of gene mentions with GNAT. *Bioinformatics*, 24(16):i126–132.

J. Hakenberg, R. Leaman, NH. Vo, S.Jonnalagadda, R. Sullivan, C. Miller, L. Tari, C. Baral, and G. Gonzalez. 2010. Efficient extraction of protein-protein interactions from full-text articles. *IEEE/ACM Trans Comput Biol Bioinform*, 7(3):481–494.

Y. Hao, X. Zhu, M. Huang, and M. Li. 2005. Discovering patterns to extract protein-protein interactions from the literature: Part II. *Bioinformatics*, 21:3294–3300.

J. Kazama, T. Makino, Y. Ohta, and J. Tsujii. 2002. Tuning support vector machines for biomedical named entity recognition. In *Proc. of BioNLP at ACL'02*, page 8.

S. Kim, J. Yoon, J Yang, and S. Park. 2010. Walk-weighted subsequence kernels for protein-protein interaction extraction. *BMC Bioinformatics*, 11(1):107.

R. Kuboyama, K. Hirata, H. Kashima, KF. Aoki-Kinoshita, and H. Yasuda. 2007. A spectrum tree kernel. *Information and Media Technologies*, 2(1):292–299.

H. Liu, V. Keselj, and C. Blouin. 2010. Biological Event Extraction using Subgraph Matching. In *SMBM'10*, October.

M. Miwa, R. Sætre, Y. Miyao, T. Ohta, and J. Tsujii. 2008. Combining multiple layers of syntactic information for protein-protein interaction extraction. In *SMBM'08*, pages 101–108.

M. Miwa, R. Sætre, Y. Miyao, and J. Tsujii. 2009. A Rich Feature Vector for Protein-Protein Interaction Extraction from Multiple Corpora. In *EMNLP'09*, pages 121–130.

S. Pyysalo, A. Airola, J. Heimonen, J. Bjrne, F. Ginter, and T. Salakoski. 2008. Comparative analysis of five protein-protein interaction corpora. *BMC Bioinformatics*, 9 Suppl 3:S6.

F. Rinaldi, G. Schneider, K. Kaljurand, S. Clematide, T. Vachon, and M. Romacker. 2010. Ontogene in biocreative ii.5. *IEEE/ACM Trans Comput Biol Bioinform*, 7(3):472–480.

L. Smith, T. Rindflesch, and W. J. Wilbur. 2004. MedPost: a part-of-speech tagger for bioMedical text. *Bioinformatics*, 20(14):2320–2321, Sep.

JM. Temkin and MR. Gilder. 2003. Extraction of protein interaction information from unstructured text using a context-free grammar. *Bioinformatics*, 19(16):2046–2053, Nov.

D. Tikk, P. Thomas, P. Palaga, J. Hakenberg, and U. Leser. 2010. A comprehensive benchmark of kernel methods to extract protein-protein interactions from literature. *PLoS Comput Biol*, 6:e1000837.

D. Zhou and Y. He. 2008. Extracting interactions between proteins from the literature. *J Biomed Inform*, 41(2):393–407, April.