Sequential Patterns for Extracting Protein-Protein Interactions from Biomedical Texts

Přemysl Vítovec, Jiří Kléma
Gerstner Laboratory
Department of Cybernetics
Czech Technical University in Prague
{vitovpre,klema}@fel.cvut.cz

Report No: GL 226/11

Abstract

This research report explores the use of sequential patterns in the task of protein-protein interaction extraction from biomedical texts. We introduce two concepts of automated sequential pattern construction and validation: frequent patterns tuning (FPT) and maximal generic patterns instantiation (MGPI). In FPT, frequent patterns are first mined from the training data, and next, after filtering out irrelevant patterns, additional patterns are derived from the mined patterns using taxonomies and linguistically sound distance constraints to improve the generalizing power and to better fit the underlying structure. The MGPI, on contrary, provides more control over the pattern structure: the pattern is structurally determined by a predefined generative grammar and only specific subsequences, common to many other related generic patterns, are extracted from the training data (and subsequently validated) independently of the current generic pattern. Moreover, addressing the essentially non-sequential character of natural language, we propose a method of text preprocessing aimed to improve the performance of arbitrary interaction extraction methods employing sequential patterns. To evaluate and compare the predictive power of both sequential pattern concepts and to determine the relevancy of the preprocessing step, we present a comprehensive set of experiments performed on standard data sets annotated for protein-protein interactions. The results rise the need for discussion about suitable testing method, mostly targeting inconsistencies in interaction annotations among and within the testing corpora.

1 Introduction

Biomedical texts contain a huge body of scientific findings written down in an unstructured way. Protein-protein interactions (PPIs) represent one of the most relevant pieces of information whose automated extraction helps to make the search for relevant information easier, improves overall understanding of biological processes and facilitates construction of their structured models. In this paper, we address the problem of identification of protein interaction pairs in texts with the aid of linguistic rules based on sequential patterns.

To put it in simple terms, we search for frequent sequences of linguistic elements/items that characterize protein-protein interaction with sufficient confidence. The linguistic rule has a form *if a sequential pattern containing two protein names then the referred proteins interact*. Obviously, the sequential pattern has to satisfy several constraints. Firstly, the pattern must contain two protein names. Secondly, the pattern must be frequent, i.e. to have a reasonable number of occurrences in the text corpora. Thirdly, we search for emerging patterns, the patterns whose frequency significantly changes from the class of protein interactions to the class of accidental protein co-occurrence. To exemplify, the sequential linguistic rule *if P1 activate P2 then P1 and P2 interact* is a rule that is likely to satisfy all the constraints as it is simple enough to be frequent and contains the verb that typically characterizes interaction. The sequential pattern *P1 activate P2* makes the conditional part of the rule which applies to any word sequence that contains two protein names and the verb to activate in the given order and within a reasonable linguistic distance (*P1 activates P2*, *P1 is activated by P2*, and possibly also to the phrase *P1 surprisingly does not activate P2* to provide an occurrence of the sequential pattern match out of the class of protein interactions).

The use of (sequential) patterns represents one of the principal approaches to the PPI extraction task. One of its main advantages lies in human understandable patterns, there are methods that enable their manual construction as well as fully automatic learning. In this report, we deal exclusively with automatic methods of sequential pattern mining and systematically study their performance on benchmark data sets annotated for protein-protein interactions in terms of precision, recall and F-measure. We address two key aspects of automated sequential pattern mining. First, we evaluate the influence of *the degree of control over the structure of target patterns*. For this purpose, we distinguish two concepts of automated sequential pattern construction: frequent pattern tuning (FPT) and maximal generic pattern instantiation (MGPI). FPT relies on a general sequential mining procedure that does not work with any structural constraints dedicated to the natural language. In brief, any frequent sequential pattern containing two protein names is extracted from the training data, further validated and generalized. MGPI, on contrary, provides more control over the pattern structure. The pattern is structurally determined by a predefined generative grammar and only specific subsequences, common to many generic patterns, are extracted from the training data (and subsequently validated) independently of the current generic pattern. Next, we change *the degree of text preprocessing* to address the essentially non-sequential character of natural language. Intuitively, sequential patterns extracted from unpreprocessed texts should exhibit low precision, since they do not cover sufficiently the underlying clause structure, even if structural restrictions like prepositional distance are applied. We introduce several preprocessing procedures to minimize the gap between expressivity of sequential patterns and the tree-like structure of natural language. Clause splitting splits clauses and minimizes extraction of patterns based on protein occurrences with no coordination. Clause linearization takes an arbitrarily complex clause of non-sequential nature and transforms it to an equivalent set of structurally simpler sequences being sequential with respect to the highest level predicate. Argument propagation reduces the effect of anaphoricity and pronominality. Text compression minimizes a sequence of nominal chunks by removal of all redundant components.

Although the experiments do not provide as unambiguous results as expected, the following conclusions can be drawn. The increased degree of control over the structure of target patterns brings a higher precision, MGPI dominates over FPT. The increased degree of text preprocessing improves FPT precision, while it has an inconclusive effect on MGPI. The previously mentioned growths in precision are compensated in recall which results in inconclusive effects on F-measure. Although there are obvious inconsistencies in interaction annotations among and within the testing corpora, the main conclusion is that the straightforward application of sequential pattern mining to the task of PPI extraction represented by the FPT method with no preprocessing performs surprisingly well.

The rest of this report is organized as follows. Section 2 motivates and overviews text preprocessing carried out in the study. Section 3 deals with automatic methods of sequential pattern mining. Section 4 describes the experimental workflow and summarizes the reached results. Finally, Section 5 provides conclusions and future work.

2 Text preprocessing

Even though every language utterance (speech, written text) shows itself as a sequence, the underlying higher order language units (texts, sentences, clauses) are tree constructs. As a result, application of sequential patterns to raw textual data leads often to false positives, as shown in examples 1 and 2, assuming the semantically relevant sequential pattern *P1 activates P2*.

(1) *A activates all these proteins and B inhibits... → interact(A, B)?*

(2) *A activates B (-)induced proteins... → interact(A, B)?*

(3) *A activates B and inhibits C → A activates B, ⟨none⟩ inhibits C*

In ex. 1, the involved proteins *A* and *B* are subjects of different clauses, i.e. there is no coordination between *proteins A* and *B*. Based on this observation, the *clause splitting* rises as a first requirement for the text preprocessing. In ex. 2, on the other hand, the involved proteins reside in the same clause, but each of them belongs to a different predication, *A* being subject of the verb predicate *activates* and *B* of the nested nominal predicate *(-)induced*. Thus, the second requirement for the text preprocessing is to convert the clause to such a form whose sequentiality dominates over the tree-like nature. This step will be referred to as *clause linearization*. Moreover, having segmentized the sentence into clauses, i.e. isolated the top level sentence predications into separated sequences, some of them may have lost one of their crucial arguments, as demonstrated in ex. 3. To prevent false negatives, one more requirement needs to be raised: *argument propagation.*

In conclusion, the text preprocessing we propose works in three subsequent steps: (1) *clause splitting*, (2) *clause linearization* and (3) *argument propagation.* In addition to that, we will also discuss a *text compression* technique aiming to remove irrelevant and noisy elements from the text. All heuristic rules introduced in the following sections are applied to texts tagged for parts-of-speech.

Methods of text preprocessing leading to structurally simpler forms have already been reported in the literature. Within the biomedical domain, MIWA ET AL. [12] apply a predefined set of linguistic rules to deep parser output; JONNALAGADDA ET AL. [10], in contrast, employ text transformations to improve the quality of the subsequent parse. A comprehensive overview of general text simplification have been presented by SIDDHARTHAN [21], who considers also discourse relations to ensure the cohesion of the preprocessed text. Text simplification has been also discussed in other application domains, such as semantic role labeling [24], headline generation [5] or sentence summarization [23].

2.1 Clause splitting

The clause splitting proceeds as detection of clausal coordination and subordination indicators. Since subordination principle is most typically restricted to clausal constructs, subjunctions (e.g. *because*) are reliable indicators of clause boundaries, however, the return from the subordinated level back to parent level is often by no means indicated. Coordination, on the other hand, appears on each level of the sentence structure, therefore the coordination indicators (i.e. commas, dashes, conjunctions) are typically ambiguous. To deal with this task, a set of heuristic rules derived from stylistic commonalities is applied.

The chunk and token level coordinations together with appositions and exemplifications are handled similarly as MIWA ET AL. reported in [12]: *A and protein* $\rightarrow$ *A* etc. However, we propose the following extension, expressed in a simple rule: if two nominal chunks both containing named entities appear coordinated and there is no reason not to treat them as such, merge them into one chunk, e. g. *A and B* $\rightarrow$ *A+B*. However, if it comes to a protein pair, both of which show strong affinity one to a preposition (i. e to the left) and one to a (nominal) predicate (i. e. to the right), they cannot be merged: e. g. *... of A and B activation of....*

Coordinated nominal chunks sequences are also frequent in biomedical literature. They are treated in the following manner: For each nominal chunk sequence, new clause is created, in which the original coordination is replaced by the particular chunk sequence. Moreover, if any subsequent coordinated chunk sequences contain protein names, they are extracted as separated sequence, e.g. *... A, inhibitor of B.*

2.2 Clause linearization

The clause linearization takes an arbitrarily complex clause (not sentence) as input and transforms it to an equivalent set of structurally simpler sequences, whose se-

[1] Structures NV of NOM by NOM (e. g. *activation of P2 by P1*) are not covered due to argument variability.

[2] Structures NA of NOM to INF (PREP) NOM (e. g. *ability of P1 to interact with P2*) are also covered.

Table 1: Nominal predicates: structure. Legend: nc ∼ noun chunk, ncs ∼ noun chunk sequence, nv ∼ verbal noun, na ∼ adjectival noun, pred ∼ predication, to ∼ *to*, by ∼ *by*, prep ∼ preposition, inf ∼ infinitive, pp ∼ past participle, adj ∼ adj.

Class	Type	Pred. scheme Left arg.	Predicate	Right arg.	Example
A	I[1]	nc	nv prep	nc/ncs/pred	A activation of B
	II[2]	nc	na to inf (prep)	nc/ncs/pred	A ability to activate B
	III	nc	pp	nc/ncs/pred	A (-)induced B
B	IV	nc/pred	pp by	nc/ncs/pred	A induced by B
	V	nc/pred	ing (prep)	nc/ncs/pred	A interacting with B
	VI	nc/pred	adj prep	nc/ncs/pred	A necessary for B

Table 2: Nominal predications as argument. Legend: nc ∼ noun chunk, ncs ∼ noun chunk sequence, nv ∼ verbal noun, na ∼ adjectival noun, pred ∼ predication, to ∼ *to*, prep ∼ preposition, inf ∼ infinitive, pp ∼ past participle

Type	Resolvent(s) generally	Example
I	nc nv nv prep nc/ncs/pred	A activation activation of B
II	nc na na to inf nc/ncs/pred	A ability ability to activate B
III	pp nc/ncs/pred	(-)induced B

quentiality with respect to the highest level predicate dominates over the underlying non-sequential nature. Thus, it partially reveals the clause structure, but without introducing common dependency or constituent tree constructs. What it actually does, is determining the extent of individual predications contained in the sentence. A predication decomposes into *predicate*, either verb predicate or one of the nominal constructs listed in Table 1, and its *left* and *right* argument areas. Instead of individual tokens, the clause is modelled as a sequence of predicates (predA, predB) and intermediate argument areas (arg), as shown in Figure 1 above.

Each clause is split into two parts: (1) left boundary + left argument area of the verb predicate and (2) verb predicate + right argument area of the verb predicate. The structure within these parts is modelled as left-to-right descending cascade, the upper level being function of the immediate lower level. This function depends on

Text	[b]	arg_1	predA_2	arg_3	verb_4	arg_5	predB_6	arg_7
		A	*activation of*	*B*	*stimulates*	*C*	*induced by*	D
Verbal	[b]		resolvent(1,2,3)		verb_4	arg_5		
			A activation		*stimulates*	*C*		
			activation of B		*stimulates*	*C*		
Nominal			arg_1 predA_2 arg_3				arg_5 predB_6 arg_7	
			A activation of B				*C induced by D*	

Figure 1: Clause linearization principle. Legend: predA, predB ∼ predicates; arg ∼ argument area; [b] ∼ boundary.

sequential ordering of the involved predicates. Predicates of class A contribute to the nearest left predication by *resolvent* (see Table 1), thus becoming directly part of the upper predication; predicates of class B, on the other hand, do not join the upper predication personally. This difference in syntactic behaviour results from the different roles these predicates play in the information structure: A-class predicates become focus of the previous predication, while B-class predicates only append an additional information to the previous content. Demonstration is given in Figure 1.

The algorithm of the clause linearization proceeds as follows:

1. The input sequence is transformed into sequence of predicates and intermediate argument areas;
2. The new sequence is further decomposed into three subsequent parts: *left verb argument area* (L), *verb predicate* (V) and *right verb argument area* (R);
3. The following procedure is applied both to L and R: Starting from the rightmost construct, the transformations listed in Table 3 are iteratively applied, each producing one *terminated* sequence, which is appended to the result pool, and one or more *unterminated* sequences, which are passed to the next iterations, until no transformation is applicable.
4. All possible sequences are composed from unterminated sequences L, V and unterminated sequences R. Results are stored in the result pool.

For simplicity reasons, we do not discuss the coordinated nominal predicates in this overview. Furthermore, note that the clause linearization is designed with the following normalization principle in mind: if it is possible, make each sequence contain at most one predicate (ex. 4), otherwise ensure that the possible argument confusion is not fatal (ex. 5).

(4) *A activates B (-)induced C* $\rightarrow$ *A activates (-)induced C, B (-)induced C*

(5) *A activates inhibition of B by C* $\rightarrow$ *A activates inhibition of B by C*

2.3 Argument propagation

Each predication takes physically place at the clause level, but all its components are not necessarily physically present in the clause, i.e. individual clauses are not independent from each other. Anaphoricity and pronominality are widely employed within sentence to build interconnections between individual clauses, thus effectively preventing the redundancy. As a result, instead of sentence in ex. 6 we most probably meet its modifications given in ex. 7.

(6) *A protein activates B and A protein also interacts with C*

(7) *A protein activates B and it also interacts with C* or *A protein activates B and also interacts with C*

Having split the sentence into individual clauses, we need to recover the semantics virtually expressed in ex. 6 from the original expressions in ex. 7. The *argument propagation*, applied to linearized clauses, proceeds by propagating the entire verb

Table 3: Transformations used in the linearization process. Legend: predA, predB $\sim$ predicates; arg $\sim$ argument area.

Action	Feature	Scheme	Example
1.	Input:	... verb|boundary$_1$ arg$_2$ predA$_3$ arg$_4$	*... activates A (-)induced B*
	Term.:	arg$_2$ predA$_3$ arg$_4$	*A (-)induced B*
	Unterm.:	... verb|boundary$_1$ resolvent$_{2,3,4}$	*... activates (-)induced B*
2.	Input:	... verb|boundary$_1$ arg$_2$ predA$_3$ arg$_4$	*... activates A induced by B*
	Term.:	arg$_2$ predA$_3$ arg$_4$	*A induced by B*
	Unterm.:	... verb|boundary$_1$ arg$_2$	*... activates A*
3.	Input:	... arg$_1$ predA$_2$ arg$_3$ predA$_4$ arg$_5$	*... A activation of B (-)induced C*
	Term.:	arg$_3$ predA$_4$ arg$_5$	*B (-)induced C*
	Unterm.:	... arg$_1$ predA$_2$ resolvent$_{3,4,5}$	*... A activation of (-)induced C*
4.	Input:	... arg$_1$ predA$_2$ arg$_3$ predB$_4$ arg$_5$	*... A activation of B induced by C*
	Term.:	resolvent$_{1,2,3}$ predB$_4$ arg$_5$	*B induced by C*
	Unterm.:	... arg$_1$ predA$_2$ arg$_3$	*... A activation of B*
5.	Input:	... arg$_1$ predB$_2$ arg$_3$ predA$_4$ arg$_5$	*... A activating B (-) induced C*
	Term.:	arg$_3$ predA$_4$ arg$_5$	*B (-)induced C*
	Unterm.:	... arg$_1$ predB$_1$ resolvent$_{2,3,4}$	*A activating (-)induced C*
6.	Input:	... arg$_1$ predB$_2$ arg$_3$ predB$_4$ arg$_5$	*... A necessary for B induced by C*
	Term.:	arg$_3$ predB$_4$ arg$_5$	*B induced by C*
	Unterm.:	... arg$_1$ predB$_2$ arg$_3$	*... A necessary for B*

argument areas to corresponding positions, either empty or purely pronominal argument areas, using the rules in Table 4. Note that the relations between clauses have been detected during the clauses segmentation phase.

The syntactic transformations described here are similar to those described by JONNALAGADDA ET AL. [10], and more extensively by SIDDHARTHAN [21]. On contrary, MIWA ET AL. do not propose such transformation, but they define rules for selecting relation related regions, either single clauses or concatenations of two subsequent clauses, one of which being relative or predicative clause. VICKREY AND KOLLER [24], on the other hand, go even further in application of syntactic transformations: the convert each clause to its canonical form.

2.4 Text compression

The notion of *head word* and *attributes*, used in linguistics to denote the internal structure of the *nominal phrase*, can be approximated within our sequential paradigm by assuming the right most nominal of the nominal chunk to be the head word while the others to be attributes. Attributes put specifications (or even restrictions) onto the head word, thus causing the sentence to diverge from its hypothetical generic meaning. In the mining process, however, these specifications are often irrelevant, as we are interested in a particular and highly specific type of information; moreover, they can even divert the mining process from crucial text components to minor ones. This can be prevented by simplifying the nominal chunks using the following rule: is there is no protein name in attributive position, replace the nominal chunk with its head word (ex. 8), otherwise propagate the protein name into the head position and replace the nominal chunk by the new head (ex. 9).

(8) *active protein A* $\rightarrow$ *A*

(9) *sudden A activation* $\rightarrow$ *A*

Table 4: Basic argument insertion. Legend: inserted left or right argument areas marked as underlined; arrival places marked with asterisk; removed parts marked in square brackets.

Clause	Example
Coor. cl.	*P1 activates P2 [and] * interacts with G3*
Subor. cl.	*P1 activates P2 [because it] * interacts with G3*
Subor. cl.	*[because] P1 appears in cells [it] * interacts with P2*
Rel. cl.	*P1 activates P2 [which] * interacts with G3*
Rel. cl.	*P1 activates P2 [which] G3 stimulates **
Rel. cl.	*P1 appears in cells [in which it] * interacts with P2*
Rel. cl.	*we investigated P1 [which] P2 binds to **
ing-cl.	*P1 stimulates P2 [by] * activating G3*
ing-cl.	*P1 stimulates P2 [,] * activating G3*
Inf. cl.	*P1 stimulates P2 [to] * activate G3*

In current grammar models, prepositional arguments play crucial role in determining relations, including those needed to detect protein interactions. On the other hand, many of them are facultative arguments of predicates or arguments of other language units; and these modifiers may again bring the confusing redundancy to the mining process. This can be prevented by applying the following two-step procedure: (1) for each nominal chunk containing a protein name and preceded by the *of*-preposition, remove both the chunks and the preposition and replace the preceding nominal chunk by the protein name; (2) for each nominal chunk not containing any protein name and preceded by a preposition, remove both the chunk and the preposition (ex. 10 and 11). This procedure (*nominal chunk sequence reduction*) is applied as the second step of sequence simplification, therefore all nominal chunks are structurally minimal.

(10) *A binds in close proximity to B* $\rightarrow$ *A binds to B*

(11) *A activated in close proximity by expression of B* $\rightarrow$ *A activated by B*

The text redundancy with respect to the particular purpose of the text mining method has been addressed multiple times in the literature: The replacement of a nominal chunk by a single word (head word) is used in the work of Jonnalagadda et al. [10]. Other authors operate with higher level language structures, leaving out peripheral phrases [5, 10], the whole clauses [12, 23] or specific semantic class of expressions such as time expressions [5].

3 Sequential patterns

Pattern based methods are widely used in the domain of the biomedical relation mining. The pattern concepts include handcrafted patterns and the patterns induced automatically from the training data. The pioneering work of Ono et al. [15] employs manually created patterns in the form of regular expressions; Blaschke and Valencia [1] use syntactic frames extracted manually from textual data and scored according to confidence and frequency. Most of the approaches, however, aim at automated pattern generation: Hakenberg et al. [9, 8] base their highly promising approach upon the idea of sentence alignment, which has been also utilized by Huang et al.. Plake et al. [16] take a predefined set of patterns as an input for genetic algorithm, designed to optimize these patterns to better fit the language data. Finally, Chiang et al. [4] and Cellier et al. [3] apply frequent patterns mining to discover relevant patterns. A comprehensive study of sequential pattern mining have been also presented by Mendes et al. [11], even though in a different text mining domain.

In this chapter, we present two concepts of sequential pattern mining: *frequent patterns tuning* (FPT) and *maximal generic patterns instantiation* (MGPI). Both concepts build upon the principle of the frequent patterns. The key difference lies in the scope in which this principle is applied or, alternatively, in whether both the pattern structure and lexical instantiations or rather the latter are learned in the learning step. Both approaches require the input text to be tagged for parts-of-speech and protein names, no predefined list of interaction expressing words is used, no human intervention is needed. Both methods involve three steps: (1) pattern generation, (2) pattern validation and (3) interaction extraction.

3.1 Frequent patterns tuning

From the strictly linguistic point of view, frequent sequential patterns are not expected to perform well in the task of relation mining from the scientific texts, even if applied to simplified, one-predicate sequences obtained in the preprocessing step. The reason is that they are derived purely from the surface structure (sequence of words). They are, in principal, unable to mirror sufficiently the underlying deep structure, which in fact is modelled as a tree construct in linguistics. However, the simplicity of this approach and the fact that a great portion of grammatical meaning is indeed expressed by the position of the particular element in the word sequence in English motivates us to investigate the possibilities of improving the predictive power of frequent patterns.

Cellier et al. [3] employ frequent patterns mining in a straightforward way: Frequent patterns are considered as candidate patterns for interaction extraction which, however, proceeds manually. To lower the number of candidate patterns, they filter the mined pattern set using simple constraints (two genes in the patterns etc.). Moreover, to give prominence to the most significant patterns, they employ recursive mining. Even though this approach does not consider any structure behind the text, the reported performance of the patterns applied to raw text is surprisingly good; though the testing method is rather unclear. In contrast, Chiang et al. stress the importance of the structural component of the patterns. They use a predefined list relation expressing words as seeds to obtain word sequences (starting and ending with protein names), which are further generalized to corresponding tag sequences. Those sequences exhibiting reasonable frequency are selected for manual curation. The reported performance of such patterns is also very good, though also here the testing method is not very clear.

The principle of FPT is to stimulate the convergence of the initial set of frequent patterns to a selection of high confidence patterns in the pattern validation process, while not significantly decreasing the potential recall of the resulting set. The tuning process operates as follows: for each pattern in the set of the mined frequent patterns, a limited set of new patterns is created by applying predefined rules which each adds a single modification to the pattern, either structural or lexical; thus, the mined frequent patterns are populated in a controlled way. In the pattern validation process, those pattern variants which capture the real language structure more reliably are expected to achieve higher confidence rate.

Structural dimension: distance constraints Each predicate binds specific arguments (semantic participants), some of which are obligatory and the other facultative. Generic patterns contain only a predicate and its obligatory arguments, but in real texts, predicates may bind also various other circumstantials, thus putting specifications or even restrictions on the respective predication. Arguments, after linearization restricted to *subject, object and prepositional* arguments, are expressed either by *nominal chunk* or by *preposition followed by nominal chunk*. However, these structures are not necessarily direct arguments of the predicate, but they bind instead to some other language unit. Most typically, the closer to the predicate, the more likely to be a direct predicate argument. This observation introduces the need of a distance measure to be integrated into the pattern mining process.

Due to the model mismatch (sequential vs. tree-like), we point out, how grammar terms are understood in the sequential paradigm, before discussing the proposed distance measure: First assume the pattern *P1 activates P2*. In this pattern, *P2* is an *object argument* according to the grammar. In the sequential paradigm, this translates to *nominal chunk containing a protein name and following the predicate in some distance*. Similarly, consider now the pattern *P1 binds to P2*. In grammar terms, *to P2* is the *prepositional argument*; in the sequential terms, this is expressed as *preposition following the predicate in some distance and nominal chunk following the preposition in some distance*.

The distance measure targeted for FPT is defined as a number of prepositions differing from *of* between two arbitrary language units. The list of controllable prepositional distances is presented in Table 5. Clearly, by setting the maximum allowable distance, we are able to align (approximately) the sequential model to the common grammar model. In the following, we discuss the particular effects of this procedure on all four controllable distances.

The *object argument* appears most typically immediately behind verbal predicate, thus, given the pattern *P1 activate P2*, setting the d_1 to zero will probably prevent the extractor from false positives, as shown in ex. 12. However, in some rare cases (ex. 13), this can also produce false negatives.

(12) *A [activates] cell growth in$_{prep}$ absence of [B], maxdist* $= 0 \Rightarrow$ *TN*[1]

[1]in square brackets elements between which the prepositional distance is investigated

Table 5: Controllable distances expressed by the prepositional distance. Legend: in square brackets elements between which the prepositional distance is investigated; vpred ∼ verb predicate, npred ∼ noun predicate, prep ∼ preposition, prot ∼ protein name, nc ∼ nominal chunk.

From	To	Examples	d
vpred	nc	*A [activates] [B]*	0
		A [activates] expression of [B]	0
vpred	prep	*A [binds] [to B]*	0
	(+ nc)	*A [binds] in close proximity [to B]*	1
prot	npred	*[A] [activation] of B*	0
		[A] [induced] B	0
prep	nc	*A necessary [for] [B]*	0
		A necessary [for] expression of [B]	0

(13) *A [activates] multiple proteins like$_{prep}$ [B], maxdist* $= 0 \Rightarrow$ *FN*

Prepositional arguments exhibit relatively floating behaviour, as a result of which the effect of setting the maximum allowable prepositional distance to a fixed value appears to be more complex. Ex. 14 and 15, assuming the pattern *P1 binds to P2*, demonstrate the positive and negative effect, respectively.

(14) *A [binds] in$_{prep}$ close proximity to$_{prep}$ other protein than$_{prep}$ [to B], maxdist* $= 1 \Rightarrow$ *TN*

(15) *A [binds] in$_{prep}$ close proximity to$_{prep}$ proteins [like B], maxdist* $= 1 \Rightarrow$ *FN*

The tight connection between nominal predicates and their left arguments is extremely stable, therefore setting the prepositional distance to minimum definitely improves the precision of the interaction extraction, as shown in ex. 16, considering the pattern *P1 activation of P2.*

(16) *... of [A] together with$_{prep}$ [activation] of B, maxdist* $= 0 \Rightarrow$ *TN*

Prepositions require their arguments to immediately follow them without exception, thus setting the prepositional distance to minimum seems reasonable to avoid false negatives, see ex. 17 employing the pattern *P1 necessary for P2.* However, counter examples do also exist (ex. 18).

(17) *A necessary [for] cell proliferation without$_{prep}$ [B contribution], maxdist* $= 0 \Rightarrow$ *TN*

(18) *A necessary [for] multiple proteins like$_{prep}$ [B], maxdist* $= 0 \Rightarrow$ *FN*

The idea of distance constraints is not new [1, 16, 18]. However, other authors understand the distance as a number of words between two elements of interest, while our definition of distance measure is motivated linguistically.

Lexical dimension: taxonomic relations The input for the sequential mining algorithm is a set of token sequences, where each token is composed of a stemmed word and a grammar tag, assigned to the word by a language tagger and possibly further refined during the text preprocessing phase. Such composits hold the most specific information about the individual words of the original sentence. The information decomposes into (1) semantic and (2) grammar component, e.g. assuming the token *activate@VBZ*, the semantics is given from the great part by the lexical meaning of the verb *activate*, whereas the grammar specification, namely third person singular present form, is held by the grammar tag.

When searching for frequent patterns or during the pattern validation phase, the specificity of the grammar component may lower the chance to find or validate less common, yet important patterns, therefore the specific grammar tags are replaced with more generic ones: all verb tags are replaced by a single tag, similarly for nouns, adjectives and prepositions. As a result, the discriminative power decreases, but in fact, we only need to differentiate between basic parts-of-speech (*tagclasses*). However, two specific situations motivate not only to follow the path of abstraction, but also to include patterns with multiple level of abstraction: (1) The output of the tagger contains misclassified words, e.g. *bind* or *interact* classified as nouns etc., thus by abstracting from the grammar specification also misclassified tokens can be matched. (2) The discriminative power of a sequential pattern without the lexical component of a selected token may be high enough in order to include only the grammar specification. Assuming that the corresponding variants with the lexical component included are infrequent, the pattern containing generalization may effectively cover all of them. However, fully lexicalized tokens should naturally represent the core of any sequential pattern.

As a result, we operate with the following three-level taxonomy: (i) *word@tagclass*, (ii) *word*, (iii) *tagclass*. Sequential patterns may combine tokens from multiple levels, though abstract levels make sense only in context of the two motivations, from which they have been derived. Irrespective to whether taxonomy is included or not, one taxonomical relation is always included: protein names are replaced by the general entity tag in all sequential patterns. Patterns with generalized tokens are derived in the postprocessing phase from the mined fully specified patterns. This approach prevents the mining process from intractable computational complexity and, in addition to that, it also corresponds to the above motivation principles.

Taxonomical relations, as introduced by Agrawal and Srikant [22], are implicitly used in any interaction extraction system combining multiple abstraction levels in their pattern architecture, e.g. Chiang et al. combine relation expressing word with part-of-speech tags; Hakenberg's multilayer sentence alignment [8] is another example, though somewhat specific.

3.2 Discussion

Intuitivelly, sequential patterns derived from frequent patterns should exhibit low precision, since they do not cover sufficiently the underlying clause structure, even if structural restrictions like prepositional distance are applied. As a demonstration, assume the pattern *P interact@VERB with@PREP P* being applied to clauses 19, 20 and 21. Clearly, it matches all the three clauses, however, two of the matches (20 and 21) generate false positives, since the negation elements *fail* and *unable* have not been taken into account.

(19) *P1 is able to interact with P2*

(20) *P1 fails to interact with P2*

(21) *P1 is unable to interact with P2*

To solve this problem, one could require only the most specific matching pattern to be considered for classification. Following our example, the pattern *P interact@VERB with@PREP P* would be rejected in favour of more specific pattern *P fail@VERB interact@VERB with@PREP P* in clause 20, similarly, in clause 21, the pattern *P1 unable@ADJ to@TO interact@VERB with@PREP P* would be preferred. However, are we likely to obtain a comprehensive set of such extended patterns by simply searching for frequent patterns? Following the path of intuition, the answer is no. More specifically, we may obtain such patterns for some of the keywords, say *interact*, but hardly for all the relevant ones, say *activate, inhibit, induce* etc.

Obviously, this is a consequence of learning the patterns as a whole which implicitly assumes syntagmatic dependence between the individual components in the pattern. However, in the real world sentences the syntagmatic relations do not necessarily imply full syntagmatic dependence. To clarify this statement, consider the variants of the pattern *P interact@VERB with@PREP P* 22, 23 and 24. The subsequences delimited by the square brackets represent the blocks between which no tight syntagmatic dependence really exists (even though there are naturally syntactic rules which determine the ordering of these blocks): both modifiers *might* and *fail to* can appear with most of the finite verb forms.

(22) *P1 [might] [interact with] P2*

(23) *P1 [fail to] [interact with] P2*

(24) *P1 [might] [fail to] [interact with] P2*

As a result, instead of the patterns as a whole it seems reasonable to learn the subsequences (subpatterns) corresponding to the individual blocks. However, this requires the pattern structure to be completely known. This motivates us to propose the following alternative approach.

3.3 Maximal generic pattern instantiation

Two crucial observations can be made using the examples 22, 23 and 24: (1) the variability of the structural blocks is significantly smaller than the variability of the parent patterns; (2) the individual structural blocks appear in many patterns. Building on these observation, the idea of the MGPI can be formulated in the following way: assuming that the parent pattern structure is completely known, a comprehensive set of all structural blocks can be easily determined and the structural blocks can be learned independently.

To demonstrate the effect of the suggested concept, consider the sentence 25. Assume that we have identified the following pattern matching this clause, *P [VERB TO] [VERB PREP] P*, but we have not observed *P fail@VERB to@TO interact@VERB with@PREP P* in the training data. However, we have observed both *fail@VERB to@TO* ($\sim$ *VERB TO*) and *interact@VERB with@PREP* ($\sim$ *VERB PREP*), therefore we have a successful match. If both blocks exhibit sufficient confidence, the protein pair *P1+P2* will be classified as interaction - even though this would be hardly the case in our example.

Transcription rules		
larg	→	x ∈ PROTEIN
rarg	→	x ∈ PROTEIN
mod1	→	x ∈ ⟨none⟩, MD
mod2	→	x ∈ ⟨none⟩, VERB TO
vpred	→	x ∈ VERB, VERB PREP
pattern	→	larg mod1 mod2 vpred rarg
Example patterns		
PROTEIN VERB PROTEIN		
PROTEIN MD VERB PROTEIN		
PROTEIN VERB TO VERB PROTEIN		
...		

Figure 2: Generating generic patterns. Legend for terminal symbols: MD ∼ modal verb, TO ∼ *to*, others are self-explaining.

(25) *P1 failed to interact with protein P2*

We now discuss the learning process. Assume we dispose of a set of generic patterns of different complexity (length). For each training sequence we proceed in the following steps:

1. the most specific generic pattern matching the example is identified (typically one, however, multiple patterns differing in structure are also possible);
2. for each structural block contained in the generic pattern, the corresponding word sequence is extracted from the training example;
3. according to the label for the identified gene pair, either true positive or false positive rate of all word sequences instantiating the structural blocks of the generic pattern is increased.

Notice that all structural blocks of a generic pattern are evaluated in the same way, regardless of the semantics of the instantiating words. Thus, assuming the example 25 as a training example and *fail to* and *interact with* as the instances of the corresponding structural blocks *VERB TO* and *VERB PREP*, *interact with* will get increment in false positive, even though it normally describes interaction. However, since *interact with* does not frequently appear with modifiers like *fail to* due to the block independence, the expected behaviour (i. e. successfully identifying interacting pairs) of *interact with* will come out in positive examples.

What remains is to clarify how the generic patterns are obtained. The need for predefined set of generic patterns is a clear disadvantage of the generic pattern instantiation. However, manual pattern definition can be avoided by employing a simple generative grammar. The generative grammar consists (in general) of the following components: terminal symbols, nonterminal symbols and transcription rules. A simple example of the adaptation of this strong computational and linguistic formalism is presented in Figure 2.

The MGPI can be regarded as an adaptation of the frequent patterns principle: Instead of mining the patterns as a whole, it is used to mine the structural blocks constituting the generic pattern. Moreover, also the prepositional distance discussed as a means of control over the pattern structure is frequently applied in the generative grammar component.

4 Experimental results

There are several annotated protein-protein interaction (PPI) corpora, which have become gold standard for evaluating protein-protein extraction corpora: AIMED [13], IEPA [6] HPRD50 [6], LLL05 [14] and BIOINFER [20], all of which are studied in detail in [19]. We chose four of these corpora to evaluate our system and added two other annotated datasets: CHARLOTTE BRUN [2] corpus and BC-PPI dataset [7]. We decided for this rather extensive testing to avoid overfitting and to get the precise idea about the real performance of the proposed methods without missing any important aspect of the problem.

A large number of PPIs are expressed by a limited set of predicates, such as *activate, inhibit, interact; activation, inhibition, interaction*, thus it is almost possible to define the list manually as some approaches do, e. g. [1]. However, the real vocabulary of predicate keywords includes also other, less frequent though equally relevant words. To enable such less frequent or less obvious words to come out into the final set, we need sufficiently rich training data. Fulltext articles obviously contain a significantly richer vocabulary than the annotated corpora introduced above (manually extracted and annotated, mainly from abstracts, limited size and rather standardized vocabulary). Similarly, also the structural variability tends to be higher in fulltext articles, which proves to be essential especially for MGPI. The pattern validation requires the training data to be annotated for PPIs. Per sentence annotations are virtually limited to those datasets used for testing. However, PPIs such as HPRD[2] contain a large amount of per article annotations (articles providing evidence for PPI). A a result, the training set is composed of sentences from 1000 randomly collected articles mentioned in HPRD, each containing at least two HGNC names[3].

The complete workflow is depicted in Figures 3 and 4. We use TREETAGGER[4] for part-of-speech tagging, GENIASS[5] for sentence splitting and DMT4SP [17] for mining mining frequent patterns (FPT learning only). Note that in the validation process we are unable to use proper confidence as we are given only per article annotations, i.e. an interaction declared in HPRD is not guaranteed to be verbally expressed in the particular sentence, from which the pair has been extracted; similarly an interaction not contained in HPRD is not necessarily false positive. Therefore, instead of proper confidence, patterns are evaluated using *pseudo-confidence*, which is calculated as if HPRD were a per sentence label, with the only exception that interaction pairs containing at least one protein not contained in HPRD are simply ignored.

Tables 6 and 7 summarize the best achieved results for FPT and MGPI, respectively, in terms of precision, recall and F-measure. To evaluate the real impact of

[2]http://www.hprd.org

[3]http://www.genenames.org

[4]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

[5]http://www-tsujii.is.s.u-tokyo.ac.jp/~y-matsu/geniass

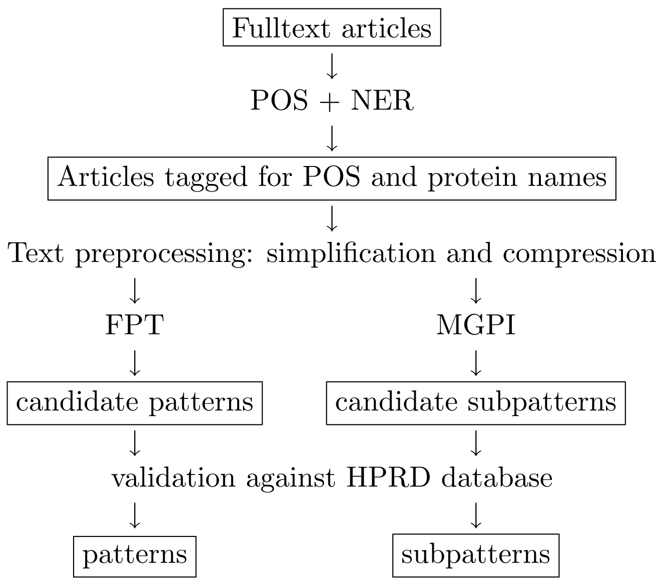
Fulltext articles
POS + NER
Articles tagged for POS and protein names
Text preprocessing: simplification and compression
FPT
MGPI
candidate patterns
candidate subpatterns
validation against HPRD database
patterns
subpatterns

Figure 3: Pattern generation workflow

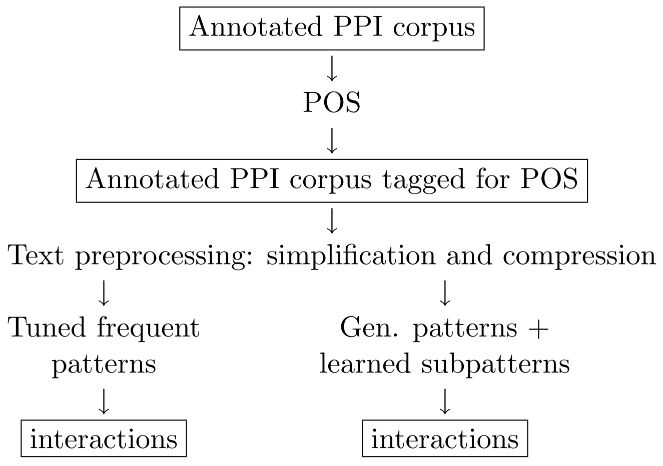
Annotated PPI corpus
POS
Annotated PPI corpus tagged for POS
Text preprocessing: simplification and compression
Tuned frequent patterns
Gen. patterns + learned subpatterns
interactions
interactions

Figure 4: Interaction extraction workflow

individual peprocessing steps, we include results for all preprocessing levels. Recall that the preprocessing techniques are not independent from each other but they are applied cummulatively, i.e. clause linearization is possible only after the sentence has been split into individual clauses and argument propagation works with linearized clauses.

The first obvious result is that FPT performs surprisingly well on the testing corpora. In accordance with our expectation, the text preprocessing (sentence simplification only, text compression will be discussed below) in its complete form leads to a higher precision rate, but as it strongly decreases the recall rate, the resulting F-measure is lower for the majority of testing datasets compared to the case when no preprocessig has been applied. In case of MGPI, the text preprocessing has lower impact on the performance rates. This is understandable, taking into account that the clause structure is to large extent reflected in the pattern structure. According to the results, the MGPI outperforms the FPT, but less than expected.

The negative impact of the text preprocessing on the recall can be explained in the follwing way: The heuristic transformations applied in the individual preprocessing steps rely to a great degree on the metalingual information contained in the part-of-speech tags, but this information is not rarely errornous. The most offending tagger errors include confusions of past tense forms (*VBD* tag) with past participle forms (*VBN* tag) and inconsistent *ing*-form disambiguation (verb, noun or adjective?). Both clause splitting and clause linearization depend strongly on the correct or at least consistent classification of these language phenomena. Further errors rise from the insufficient interpretation of various correctly tagged constructs, the most difficult being again the correct understanding of forms that are grammatically homonymous.

The effect of text compression is clearly positive. Reducing the nominal chunks to minimal chunks adds up to 6% to the F-measure in case of FTP and 1% in case of MGPI. Similarly, shortening the nominal chunk sequences increases the F-measure by 7% and 1%, respectively. Thus, the FPT benefits from text compression more strongly than the MGPI. Moreover, the text compression drastically reduces the computational time in case of the FPT. Furthermore, both pattern tuning techniques (i.e. distance constraints and taxonomies) contribute significantly to the overall performance: the F-measure of pure frequent patterns is up to 9% lower than that of FPT given in Table 6.

The testing corpora differ from each other significantly both in the range of language expressions regarded as interaction evidence and in the quality of annotations, thus leading to inconsistencies among the individual corpora and in the individual corpora. In case of CHARLOTTE BRUN corpus, we had to analyze all sentences manually, removing at least the most obvious inconsistencies. This raises the need for a serious discussion about the appropriate testing method.

Table 6: FPT: best results with (no compression). Legend: NP ∼ no preprocessing, CS ∼ clause splitting, CL ∼ clause linearization, AP ∼ argument propagation.

Corpus	Aimed			Brun		
Level	P	R	F	P	R	F
NP	0.55	0.55	0.55	0.5	0.88	0.64
CS	0.58	0.47	0.52	0.45	0.53	0.49
CL	0.59	0.43	0.5	0.36	0.52	0.43
AP	0.59	0.53	0.56	0.43	0.51	0.47
Corpus	HPRD50			IEPA		
Level	P	R	F	P	R	F
NP	0.64	0.94	0.76	0.62	0.94	0.75
CS	0.64	0.7	0.67	0.67	0.7	0.68
CL	0.54	0.62	0.58	0.64	0.56	0.6
AP	0.55	0.71	0.62	0.67	0.65	0.66
Corpus	LLL05			BC-PPI		
Level	P	R	F	P	R	F
NP	0.51	0.97	0.67	0.25	0.84	0.39
CS	0.73	0.7	0.71	0.44	0.38	0.41
CL	0.7	0.47	0.56	0.43	0.37	0.4
AP	0.71	0.6	0.65	0.43	0.52	0.47

Table 7: MGPI: best results with (no compression). Legend: NP ∼ no preprocessing, CS ∼ clause splitting, CL ∼ clause linearization, AP ∼ argument propagation.

Corpus	Aimed			Brun		
Level	P	R	F	P	R	F
NP	0.4	0.71	0.51	0.53	0.74	0.62
CS	0.53	0.5	0.51	0.65	0.39	0.49
CL	0.5	0.45	0.47	0.56	0.33	0.41
AP	0.5	0.56	0.53	0.55	0.41	0.47
Corpus	HPRD50			IEPA		
Level	P	R	F	P	R	F
NP	0.7	0.78	0.74	0.66	0.82	0.73
CS	0.76	0.59	0.67	0.81	0.57	0.67
CL	0.68	0.55	0.61	0.76	0.45	0.57
AP	0.78	0.59	0.67	0.77	0.54	0.64
Corpus	LLL05			BC-PPI		
Level	P	R	F	P	R	F
NP	0.52	0.92	0.66	0.3	0.76	0.43
CS	0.78	0.63	0.7	0.41	0.51	0.46
CL	0.74	0.48	0.58	0.51	0.44	0.47
AP	0.77	0.61	0.68	0.5	0.52	0.51

5 Conclusion and further work

The expectations raised about frequent pattern performance and the impact of the text preprocessing have not been confirmed unambiguously. Nevertheless, this does not suspend valuable conclusions: The increased degree of control over the pattern structure draws the performance towards higher precision, thus MGPI as a limiting case consistently outperforms FPT. However, a significant decrease of recall diminish the effect of the text preprocessing to the resulting F-measure. Both text compression and the proposed tuning methods exhibit clearly positive effect on the performance of the frequent patterns, as a result of which FPT performs surprisingly well, compared to MGPI, especially when no preprocessing has been applied. In the future work, we need to perform a more detailed analysis of errors to precisely identify the factors in the text preprocessing contributing to low recall rates.

Acknowledgment

The work of Přemysl Vítovec was funded by the Grant Agency of the Czech Technical University in Prague, grant No. SGS11/126/OHK3/TT/13. The work of Jiří Kléma was funded by the Czech Ministry of Education in the framework of the research programme Transdisciplinary Research in the Area of Biomedical Engineering II, MSM 6840770012.

References

[1] Christian Blaschke and Alfonso Valencia. The frame-based module of the suiseki information extraction system. *IEEE Intelligent Systems*, 17:14–20, March 2002.

[2] Christine Brun. *Christine Brun Corpus.* http://www.biocreative.org/accounts/login/?next=/resources/. Accessed March 2011.

[3] Peggy Cellier, Thierry Charnois, and Marc Plantevit. Sequential Patterns to Discover and Characterise Biological Relations. In *Lecture Notes in Computer Science*, volume 6008/2010, pages 537–548, 2010.

[4] Jung-Hsien Chiang, Hsiao-Sheng Liu, Shih-Yi Chao, and Cheng-Yu Chen. Discovering gene-gene relations from sequential sentence patterns in biomedical literature. *Expert Syst. Appl.*, 33:1036–1041, 2007.

[5] Bonnie Dorr, David Zajic, and Richard Schwartz. Hedge trimmer: A parse-and-trim approach to headline generation, 2003.

[6] Katrin Fundel, Robert Küffner, and Ralf Zimmer. RelEx—Relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

[7] Jörg Hakenberg. *BC-PPI Corpus.* Humboldt-Universität zu Berlin - Institut für Informatik, http://www2.informatik.hu-berlin.de/ hakenber/corpora/. Accessed March 2011.

[8] Jrg Hakenberg, Ha Vo Leaman, Robert amd Nguyen, Siddhartha Jonnalagadda, Ryan Sullivan, Christopher Miller, Chitta Baral, and Graciela Gonzalez. Efficient extraction of protein-protein interactions from full-text articles. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(3):481–494, 2010.

[9] Jrg Hakenberg, Conrad Plake, Loic Royer, Hendrik Strobelt, Ulf Leser, and Michael Schroeder. Gene mention normalization and interaction extraction with context models and sentence motifs. *Genome Biology*, 9:S14, 2008.

[10] Siddhartha Jonnalagadda, Luis Tari, Jorg Hakenberg, Chitta Baral, and Graciela Gonzalez. Towards effective sentence simplification for automatic processing of biomedical text. *CoRR*, pages –1–1, 2010.

[11] Ana Cristina Mendes and Cludia Antunes. Pattern mining with natural language processing: An exploratory approach. In *Machine Learning and Data Mining in Pattern Recognition*, pages 266–279, 2009.

[12] Makoto Miwa, Rune Saetre, Yusuke Miyao, and Jun'ichi Tsujii. Entity-focused sentence simplification for relation extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 788–796, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[13] Raymond J. Mooney. *AiMed.* University of Texas at Austin, https://wiki.inf.ed.ac.uk/TFlex/AiMed. Accessed March 2010.

[14] C. Nédellec. Learning Language in Logic - Genic Interaction Extraction Challenge. In *Proceedings of the 4th Learning Language in Logic Workshop 2005*, 2005.

[15] Toshihide Ono, Haretsugu Hishigaki, Akira Tanigami, and Toshihisa Takagi. Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, 17(1):155–161, 2001.

[16] Conrad Plake, Jörg Hakenberg, and Ulf Leser. Optimizing syntax patterns for discovering protein-protein interactions. In *Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, pages 195–201, New York, NY, USA, 2005. ACM.

[17] M. Plantevit, T. Charnois, J. Klema, C. Rigotti, and B. Cremilleux. Combining sequence and itemset mining to discover named entities in biomedical texts: A new type of pattern. *International Journal of Data Mining, Modelling and Management*, 1:119–148, 2009.

[18] D. Proux, F. Rechenmann, and L. Julliard. A pragmatic information extraction strategy for gathering data on genetic interactions. In *Proceedings of the 9th International Conference on Intelligent Systems for Molecular Biology (ISMB-2001)*, pages 279–85, 2000.

[19] Sampo Pyysalo, Antti Airola, Juho Heimonen, Jari Björne, Filip Ginter, and Tapio Salakoski. Comparative analysis of five protein-protein interaction corpora. *BMC Bioinformatics*, 9(Suppl 3):S6, 2008.

[20] Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Bjorne, Jorma Boberg, Jouni Jarvinen, and Tapio Salakoski. BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(1):50+, 2007.

[21] Advaith Siddharthan. Syntactic ssimplification and text cohesion. *Language and Computation*, 4:77–109, 2006.

[22] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, pages 3–17, 1996.

[23] Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Inf. Process. Manage.*, 43:1606–1618, November 2007.

[24] David Vickrey and Daphne Koller. Sentence simplification for semantic role labeling. In *Meeting of the Association for Computational Linguistics*, pages 344–352, 2008.