Recursive Sequence Mining to Discover Named Entity Relations

Peggy Cellier[1], Thierry Charnois[1], Marc Plantevit[2], and Bruno Crémilleux[1]

[1] Université de Caen, GREYC, CNRS, UMR6072, F-14032, France
firstname.lastname@info.unicaen.fr
[2] Université Lyon 1, LIRIS, UMR5205, F-69622, France
marc.plantevit@liris.cnrs.fr

Abstract. Extraction of named entity relations in textual data is an important challenge in natural language processing. For that purpose, we propose a new data mining approach based on recursive sequence mining. The contribution of this work is twofold. First, we present a method based on a cross-fertilization of sequence mining under constraints and recursive pattern mining to produce a user-manageable set of linguistic information extraction rules. Moreover, unlike most works from the state-of-the-art in natural language processing, our approach does not need syntactic parsing of the sentences neither resource except the training data. Second, we show in practice how to apply the computed rules to detect new relations between named entities, highlighting the interest of hybridization of data mining and natural language processing techniques in the discovery of knowledge. We illustrate our approach with the detection of gene interactions in biomedical literature.

Keywords: Sequential Data, Recursive Mining, Named Entity Relations, Pattern Discovery, Natural Language Processing.

1 Introduction

Due to the explosion of available textual data, text mining and Information Extraction (IE) from texts have become important topics of study in recent years. In particular, detection of relations between named entities is a challenging task to automatically discover new relationships in texts. The detection of gene interactions in biomedical texts and the discovery of companies relations (sell/buy) in newspapers belong to the scope of that problem. Some previous works use handcrafted linguistic IE rules for that task which is time consuming [7,5]. Other methods based on Machine Learning (ML) techniques [10] give good results but run as a "black box": their outcomes are not really understandable by a user and Natural Language Processing (NLP) cannot fully take benefit from them.

A key idea of this paper is to propose a cross-fertilization of data mining and NLP techniques to profit from the advantages of the two fields. More precisely, we propose a method based on *recursive sequential pattern mining* with constraints coming from the NLP field to tackle the problem of the discovery of named entity relations. Sequence mining, in particular sequential pattern mining [1], is a well-known data mining technique that allows to extract regularities in a sequence database. The recursive pattern mining [4] and the constraint-based paradigm [14] enable to give prominence to the most significant patterns. As we will see (Section 2), these techniques have suitable properties for our goal.

The contribution of this work is twofold. First, we propose a recursive pattern mining approach based on constrained sequential patterns to produce a user-manageable set of linguistic IE rules. The recursive sequential pattern mining enables the effective mastering of the number of discovered patterns that are returned. If recursive mining has already been used in the context of itemsets (i.e., data described by items), we show how such a method can deal with sequential data and we prove new properties on recursive pattern mining on sequences. Second, we show how to apply sequential patterns as linguistic IE rules to detect new relations in texts. We illustrate our approach with the detection of gene interactions in biomedical literature. With regards to the issue of the detection of named entity relations, the main advantages of our approach with respect to existing ones are that the results are automatically discovered and easily understandable by a human. To the best of our knowledge, it is the first approach combining sequential pattern mining methods and linguistic information.

The paper is organized as follow. Section 2 presents the background in data mining and existing approaches that tackle the problem of the detection of named entity relations. Section 3 gives the main contributions of this paper. Especially, we describe the discovery of linguistic IE rules to detect relations between entities. Section 4 presents a case study about IE rules for gene interaction detection.

2 Background

This section provides the background on sequential pattern mining and recursive mining. Related work on named entity relation detection is then presented.

2.1 Sequential Pattern Mining under Constraints

Sequential pattern mining [1] is a data mining technique that aims at discovering correlations between events through their order of appearance. Sequential pattern mining is an important field of data mining with broad applications (e.g., biology, marketing, security) and there are many algorithms to extract frequent sequences [19,15,22].

In the context of sequential patterns extraction, a *sequence* is an ordered list of distinct literals called *items*. A sequence S is denoted by $\langle i_1 i_2 \dots i_n \rangle$ where i_k, $1 \leq k \leq n$, is an item. Let $S_1 = \langle i_1 i_2 \dots i_n \rangle$ and $S_2 = \langle i'_1 i'_2 \dots i'_m \rangle$ be two sequences. S_1 is *included* in S_2 if there exist integers $1 \leq j_1 < j_2 < \dots < j_n \leq m$ such that $i_1 = i'_{j_1}$, $i_2 = i'_{j_2}$, …, $i_n = i'_{j_n}$. S_1 is called a *subsequence* of S_2. S_2 is called a *super-sequence* of S_1, denoted by $S_1 \preceq S_2$. An extracted sequential pattern, S_1, is *maximal* if there is no other extracted sequential pattern, S_2, such that $S_1 \preceq S_2$. A sequence database SDB is a set of tuples (sid, S) where sid is a sequence ID and S a sequence. A tuple (sid, S) *contains* a sequence T, if T is a subsequence of S. The *support* of a sequence T in a sequence database SDB is the number of tuples in the database containing T: $sup_{SDB}(T) = |\{(sid, S) \in SDB | (T \preceq S)\}|$ where $|A|$ represents the cardinality of set A[1]. Note that we do not mention the database when it is clear from the context: $sup(T)$. In this paper, we use sequences of items for sake of simplicity and because it is enough for the case study. However, this approach can be straightforwardly generalized to sequences of itemsets.

The constraint-based pattern mining framework is a powerful paradigm to discover new highly valuable knowledge [13]. Constraints provide a focus on the most promising knowledge by reducing the number of extracted patterns to those of potential interest for the user. More precisely, constraint-based mining task selects all the sequential patterns included in SDB and satisfying a predicate which is called *constraint*. There are a lot of constraints to evaluate the relevance of sequential patterns. The most well-known example is the frequency constraint. Given a minimum support threshold *minsup*, the problem of frequent sequential pattern mining is to find the complete set of sequential patterns whose support is greater than or equal to *minsup*. There are many other constraints highlighting the best sequential patterns with respect to the user objectives [14]. In this work, we will see that we use both syntactic constraints and constraints coming from linguistic information.

2.2 Recursive Pattern Mining

Recursive pattern mining [4] is a process that gives prominence to the most significant patterns and filters the specific ones. The key idea of recursive pattern mining is to repeat the pattern mining process on the output in order to reduce it until few and significant patterns are obtained. That recursive process is ended when the result becomes stable. The final recursive patterns bring forward information coming from each mining step. More precisely, a recursive pattern produces a k-summary (i.e., a set with at most k patterns) summarizing the data according to a measure (e.g., frequency, growth rate) where k is a given number. One of the advantages of recursive pattern mining is that the number of returned frequent patterns is well mastered. If recursive mining is already used with item data [4], to the best of our knowledge, we propose here the first use on sequential data. Such a use requires to demonstrate properties on sequential data and Section 3.5 is devoted to this task.

2.3 Related Work

Several approaches have been widely applied to extract knowledge from texts: NLP, in particular information extraction, and ML.

IE methods need linguistic resources such as grammars. That kind of approaches apply linguistic IE rules to extract information [7,5]. However, the

[1] The relative support is also used: $sup_{SDB}(T) = \frac{|\{(sid,S)\ s.t.\ (sid,S) \in SDB \wedge (T \preceq S)\}|}{|SDB|}$.

resources are very often handcrafted. Those methods are thus time consuming and very often devoted to specific corpus. In contrast, ML methods, for example support vector machines or conditional random fields [10], are less time consuming than IE methods. They give good results but they need many features and their outcomes are not really understandable by a user and not usable in NLP systems as linguistic patterns.

A good trade-off is a combination of IE and ML techniques which aims at automatically learning the linguistic IE rules [12,18]. However in most cases the learning process is done with a syntactic parsing of the text (shallow parsing or deep parsing). Therefore, the quality of the learned rules is relied on results of syntactic process which is currently not often a reliable process. Unlike those methods, our proposed approach does not need syntactic parsing of the sentences neither resource except the training data.

Some works [8] do not use syntactic parsing and learn surface patterns using sequence alignment of sentences to derive "motifs". One drawback of that approach is that the sequence alignment implies that patterns are learned with contiguous words. An inexact matching is nevertheless used to apply the patterns on the application corpus. Other works [9] implicitly uses sequence mining in order to compute information extraction rules. However, the number of patterns (i.e., IE rules) is not well mastered and thus they cannot be presented to an expert. Using *n-grams* is another technique very widespread to automatically extract patterns. The drawbacks of n-grams is that the size of the extracted patterns is set for all patterns to n and the elements in patterns must be contiguous. Moreover n-gram can be seen as a specific instance of sequential pattern. Unlike n-grams, in sequential pattern mining, discovered patterns can have different sizes, and items within sequential patterns are not necessarily contiguous.

3 Recursive Sequence Mining to Discover Named Entity Relations

This section presents the discovery of patterns as linguistic IE rules to detect relations between named entities in texts. First, an overview of the approach is given. Second, the use of sequential pattern mining for the detection of named entity relations is explained. Third, linguistic constraints are discussed. Fourth, the algorithm is presented. Finally properties about the recursive pattern mining step are proven.

3.1 Overview

The main idea of our approach is to extract the frequent patterns satisfying user-defined constraints. As the order of words in texts is important, our approach is based on sequential pattern mining which aims at discovering correlations between events through an ordered relation. The order of words within the sentence corresponds to the ordered relation to supply sequential pattern mining. Some linguistic constraints (cf. Section 3.3) are used in order to drive the mining process towards the user objectives. Even if the number of produced patterns is reduced thanks to the constraint, the output still remains too large for individual and global analysis by the end-user. That is why a recursive pattern mining step is performed in order to give prominence to the most significant patterns and to control the output size. That step needs that important properties on recursive pattern mining (cf. Section 3.5) are proven. In addition to usual evaluations by using Precision and Recall measures, Section 4 shows that this small number of computed sequential patterns enables a validation by linguists.

3.2 Sequential Pattern Mining for Named Entity Relations

For the extraction of sequential patterns as linguistic IE rules, the database is built from texts which contain relations and where the named entities are identified and replaced by the specific item *Named_Entity*. In order to avoid problems introduced by the anaphoric structures [23], we consider sequences containing a relation, i.e. a verb or a noun, and at least two named entities.

The choice of the support threshold *minsup* is a well-known issue in data mining. We note in our applications that some interesting words for named entity relation detection are not very frequent so that we set a low value of *minsup*. As a consequence, a huge set of patterns is discovered and it needs to be filtered in order to return only interesting and relevant patterns.

3.3 Linguistic Constraints

In pattern mining, the constraints allow to precisely define the user interest. The most commonly used constraint is the constraint of frequency (*minsup*) because it satisfies suitable mining properties. However, it is possible to use different constraints in conjunction to the frequency [13]. In our work, we use mainly two linguistic constraints on sequential patterns to discover named entity relations.

The first constraint is that the pattern must contain two named entities (C_{2ne}). The set $SAT(C_{2ne})$ represents the set of patterns that satisfy C_{2ne}: $SAT(C_{2ne}) = \{S = \langle i_1 i_2 \ldots i_m \rangle \mid \; |\{j \mid j \in 1 \ldots m \wedge i_j = \mathit{Named\_Entity}\}| \geq 2\}$. Indeed, the targeted relation is between at least two named entities.

The second constraint is that the pattern must contain a verb or a noun (C_{vn}) in order to express a named entity relation. The set $SAT(C_{vn})$ represents the set of patterns that satisfy C_{vn}: $SAT(C_{vn})=\{S=\langle i_1 i_2 \ldots i_m\rangle \mid \exists j, verb(i_j) or noun(i_j)\}$ where $verb(i_j)$ (resp. $noun(i_j)$) is a predicate that returns true if i_j is a verb (resp. noun).

Note that other linguistic constraints can be added to give more precision with respect to the kind of searched relations. In the following, for the sake of clarity, all constraints are grouped in only one constraint $\mathcal{C}_G$ and $SAT(C_G)$ is the set of patterns satisfying C_G. From the constraint-based paradigm, the C_{2ne} and C_{vn} constraints belong to the category of regular expression constraints introduced by [6].

3.4 Algorithm

Algorithm 1 presents the whole process to discover named entity relations. Firstly, the text is POS tagged (Step 1), i.e., each word is replaced by its lemma and linguistic informations. That step defines the items of the sequence database. The POS tagged text is then sliced in sequences (Step 2). The type of slice size (a sequence) can be for example the phrase, the whole sentence or the paragraph.

Sequential pattern mining is then applied (Step 3) to find the frequent sequential patterns in the database. The patterns are then filtered with respect to user-defined constraints (Step 4). Method *CheckConstrainsts* prunes the sequential patterns that do not satisfy $\mathcal{C}_G$. Therefore, the *constrainedPatterns* set contains all frequent sequential patterns that satisfy $\mathcal{C}_G$. In order to avoid the redundancy between patterns, only maximal patterns (cf. Section 2.1) are kept (Step 5). The computation of the maximality is done in post processing because it is not time consuming. It takes less than 2 minutes. However that phase can be done in the sequential pattern mining step (Step 3).

Even if the new set of sequential patterns, *maximalConstrainedPatterns*, is significantly smaller than the complete set *sequencePatterns*, it can still be too large to be analyzed and validated by a human user. Therefore we use *recursive pattern mining* [4] to filter very specific patterns. As we are interested to keep some patterns for each relation expression, i.e., for each verb or noun, X_i, the set *maximalConstrainedPatterns* is thus divided into several subsets $S(X_i)$ (Step 6)[2]. A subset $S(X_i)$ is the set of all sequential patterns of *maximalConstrainedPatterns* containing the item X_i. More formally, $S(X_i) = \{S \in \mathit{maximalConstrainedPatterns} \mid \langle X_i \rangle \preceq S\}$. Note that X_i are elements labeled as a verb or a noun. The most k ($k > 1$) representative elements for each $S(X_i)$ are then computed. Each subset $S(X_i)$ is then recursively[3] mined with a support threshold, min_sup_R, equal to $max\{\frac{|S(X_i)|}{k}, 2\}$ in order to extract frequent sequential patterns satisfying $\mathcal{C}_G$ (Steps 7–14). It means that the extracted sequential patterns become the sequences of the new database to mine. That process ends when the number of extracted patterns is less than or equal to k. Some properties about the recursive pattern mining step are given in Section 3.5.

At the end of the process, the number of sequential patterns is well-mastered. Indeed, recursive mining goes on until the number of sequential patterns is less than or equal to k, and as recursive mining always stops (Theorem 1), the number of sequential patterns for each subset is thus less than or equal to k. Therefore the number of returned sequential patterns is less than or equal to $n \times k$ where n is the number of subsets $S(X_i)$ in $SAT(\mathcal{C}_G)$. Note that k is set *a priori* by the user so that sequential patterns can be analyzed by a human. The sequential patterns are then validated by the user and considered as linguistic IE rules for the detection of relations between named entities.

[2] The $S(X_i)$ are not necessarily disjoint.

[3] The recursive process is given in iterative writing in Agorithm 1.

Algorithm 1. Discovery of Named Entity Relation Patterns

```
Input: $text$: text ; $minsup$: support threshold ; $slice\_type$: scope of a sequence in text;
    $\mathcal{C}_G$: constraints ; k: recursive mining threshold
Output: $patterns$, set of returned frequent sequential patterns
Method:
 1: $POSTaggedText := POS\_Tagging(text)$
 2: $textSDB := Slicing(POSTaggedText,\ slice\_type)$
 3: $frequentSequentialPatterns := SequenceMining(textSDB,\ minsup)$
 4: $constrainedPatterns := CheckConstrainsts(frequentSequentialPatterns,\ \mathcal{C}_G)$
 5: $maximalConstrainedPatterns := Maximal(constrainedPatterns)$
 6: $patternSets := Split(maximalConstrainedPatterns)$
 7: $patterns := \emptyset$
 8: for all $S(X_i) \in patternSets$ do
 9:    while $|S(X_i)| > k$ do
10:       $min\_sup_R := max\{\frac{|S(X_i)|}{k}, 2\}$
11:       $FP := SequenceMining(S(X_i),\ min\_sup_R)$
12:       $CP := CheckConstrainsts(FP)$
13:       $S(X_i) := Maximal(CP)$
14:    end while
15:    $patterns := patterns \cup S(X_i)$
16: end for
```

3.5 Properties of Recursive Pattern Mining

In this section, important properties about recursive pattern mining are demonstrated. These properties are new in the context of sequential data.

The first property is about the frequency of the returned patterns. For each subset $S(X_i)$, the k or less extracted sequential patterns returned after recursive pattern mining are frequent in the sequence database *textSDB*. In other words, they belong to the complete set of frequent sequential patterns in *textSDB*, *frequentSequentialPatterns*, with respect to *minsup* (Property 1).

Property 1. Let $S(X)$ be a set of frequent sequential patterns in *textSDB* that contain X. The sequential patterns of $S(X)$ after recursive pattern mining are frequent in *textSDB* with respect to *minsup.*

Proof. The proof is conducted recursively on the number of recursive pattern mining steps.

Base case: No recursive pattern mining. All elements of $S(X)$ are frequent in *textSDB* with respect to *minsup* (Step 3 of Algorithm 1).

Hypothesis: We assume that after j recursive pattern mining steps all elements of $S(X)$ are frequent in *textSDB* with respect to *minsup*. Let $S_j(X)$ be the set of sequential patterns after j recursive pattern mining steps.

Recursive Case: Let $S_{j+1}(X)$ be the set of sequential patterns after $j+1$ recursive pattern mining steps and $p \in S_{j+1}(X)$. It implies that p is frequent in $S_j(X)$ and thus there exists at least one element in $S_j(X)$, e, such that $p \preceq e$. The pattern e is frequent in *textSDB* (recursive hypothesis). Thanks to the anti-monotonicity of the support, p is thus also frequent in *textSDB*. □

We prove that the recursive pattern mining stops (Theorem 1). Unfortunately, the proof cannot be based on the strictly decreasing of the number of patterns during the recursive pattern mining, because that number may not decrease for some steps. That is why we base the proof according to the size of the largest maximal frequent sequential patterns because it strictly decreases during the recursive pattern mining (Property 2).

Property 2. Let $S_j(X)$ be the result set of sequential patterns recursively mined at step j of the recursive pattern mining (Steps 7-14) and $S_{j+1}(X)$ the result set at step $j+1$ then: $\forall p \in S_{j+1}(X) \;\; |p| < max\{|p_l| \mid p_l \in S_j(X)\}$
where $|p|$ is the size of the pattern, i.e., the number of items in p.

Proof. Let p be an element of $S_{j+1}(X)$: $p \in S_{j+1}(X)$. Let E_p be the set of all elements of $S_j(X)$ such that: $\forall e \in E_p \; p \preceq e$.

The support threshold for the sequential mining (Step 11) is greater than or equal to 2 (Step 10) and p is frequent in $S_j(X)$. It implies that $|E_p| \geq 2$. However, the elements of $S_j(X)$ are maximal (Step 13), so that the elements of E_p are also maximal (indeed $E_p \subseteq S_j(X)$) and thus p cannot be equal to any element of E_p: $\forall e \in E_p \; p \precneqq e$ (i.e., $p \preceq e$ and $p \neq e$) $\Rightarrow \forall e \in E_p \; |p| < |e|$ and $\forall e \in E_p \; |e| \leq max\{|p_l| \mid p_l \in S_j(X)\} \Rightarrow |p| < max\{|p_l| \mid p_l \in S_j(X)\}$ □

Theorem 1. *Let k be an integer, $k > 1$. The recursive pattern mining of $S(X)$ stops (cf Algorithm 1).*

Proof. The proof is conducted recursively on the size of patterns of $S(X)$.

Base case: the size of patterns of $S(X)$ is 0. The number of patterns in $S(X)$ is thus 0. In addition, $0 < k$ and thanks to Step 9 of Algorithm 1, the recursive pattern mining stops.

Hypothesis: We assume that the recursive pattern mining stops when the size of patterns of $S(X)$ is lower than or equal to T.

Recursive Case: the size of patterns of $S(X)$ is lower than or equal to $T+1$. Thanks to Property 2, after one application of the recursive pattern mining on $S(X)$ we know that $S(X)$ contain patterns such that their size is lower than or equal to T. The recursive pattern mining thus stops thanks to the recursive hypothesis. □

4 Case Study: Discovery of Gene Interaction Patterns

This section presents the discovery of frequent sequential patterns as linguistic information extraction rules for gene interaction detection. The named entities are the genes. Experiments are conducted on texts from biological and medical literature. A linguistic analysis of this case study is given in [3].

4.1 Training Dataset

We merge two different corpora containing genes and proteins to build the training dataset. The first corpus contains sentences from PubMed abstracts, annotated by Christine Brun. It contains 1806 annotated sentences. That corpus is available as a secondary source of learning tasks "Protein-Protein Interaction Task (Interaction Award Sub-task, ISS)" from BioCreAtIvE Challenge II [10]. The second corpus contains sentences of interactions between proteins annotated by an expert. That dataset, containing 2995 sentences with gene interactions, is described in [16]. The training corpus thus contains 4801 sentences.

A POS tagging is then performed on the merged corpus using the *treetagger* tool [17]. The sentences are then split into sequences to build the database. For example, let us consider two sentences that contain gene interactions:

– "*Here we show that <Gene SOX10>, in synergy with <Gene PAX3>, strongly activates <Gene MITF > expression in transfection assays.*"
– "*The <Gene Menin>-<Gene JunD> interaction was confirmed in vitro and in vivo.*"

Those sentences generate two sequences[4]:

– ⟨ *here@rb we@pp show@vvp that@in/that Named_Entity ,@, in@in synergy@nn with@in Named_Entity ,@, strongly@rb activate@vvz Named_Entity expression@nn in@in transfection@nn assay@nns .@sent* ⟩
– ⟨ *the@dt Named_Entity -@: Named_Entity interaction@nn be@vbd confirm@vvn in@in vitro@nn and@cc in@in vivo@rb .@sent* ⟩

The gene names, i.e., the named entities, are replaced by a specific item, *Named_Entity*, and the other words are replaced by the combinations of their lemma and their POS tag. The order relation between items in a sequence is the order of words within the sentence. For experiments, the sequences of the database are the sentences where each word is replaced by the corresponding item. It means that *slice_type* is the whole sentence.

4.2 Recursive Sequential Pattern Mining

For the sequential pattern mining, we set a support threshold *minsup* equal to 10. With that threshold some irrelevant patterns are not taken into account while many patterns of gene interactions are discovered. We conducted other experiments with greater *minsup* values (15 and 20). With those thresholds some relevant patterns for interaction detection are lost. The number of frequent sequential patterns that are extracted is high. More than 32 million frequent sequential patterns are discovered. Although the number of extracted patterns is high the extraction of all frequent patterns spends only 15 minutes. The extraction tool is *dmt*4 [11].

The application of constraints significantly reduces the number of sequential patterns. Indeed, the number of sequential patterns satisfying the constraints is about $65,000$. However, that number is still prohibitive for analysis and validation by a human expert. Recall that the application of constraints is not time consuming (couple of minutes).

[4] $'rb'$, $'pp'$, ... after $'@'$ are tags given by *treetagger*, for example : $'rb'$ means *adverb*, $'pp'$ means *personal pronoun.*

The sequential patterns, which are computed in the previous step, are divided into several subsets. The recursive pattern mining of each subset exhibits at most k sequential patterns to represent that subset. In this experiment, we set the parameter k to 4. We get 515 subsets (365 for nouns, 150 for verbs). At the end of the recursive pattern mining, there remain 667 candidate sequential patterns that represent interactions. That number, which is significantly smaller than the previous one, guarantees the feasibility of an analysis of those patterns as linguistic IE rules by an expert. The recursive pattern mining of those subsets is not time consuming. It takes about 2 minutes.

The 667 remaining sequential patterns were analyzed by two users. They validated 232 sequential patterns for interaction detection in 90 minutes. It means that 232 sequential patterns represent several forms of interactions between genes. Among those patterns, some explicitly represent interactions. For example, ⟨*Named_Entity deplete@vvn Named_Entity .@sent*⟩, ⟨*activation@nn of@in Named_Entity by@in Named_Entity .@sent*⟩ or ⟨*Named_Entity be@vbd inhibit@vvn by@in AGENE@np .@sent*⟩ describe well-known interactions (inhibition, activation). Other patterns represent more general interactions between genes, meaning that a gene plays a role in the activity of another gene for instance ⟨*Named_Entity involve@vvn in@in Named_Entity .@sent*⟩ or ⟨*that@in/that Named_Entity play@vvz role@nn in@in Named_Entity .@sent*⟩. Most of remaining patterns represent modalities or biological context.

The validated sequential patterns are linguistic IE rules that can be used on biomedical texts to detect interactions between genes. Note that the application of those patterns do not need a syntactic analysis of the sentence.

4.3 Detection of Gene Interactions

Following the case study, we have conducted some experiments in order to evaluate the quality of the sequential patterns found in the previous section from a quantitative point of view. For that purpose, we consider three sets of data well-known in literature: *GeneTag* from the data set *Genia* [20], *BioCreative* from [21] and *AIMed* from [2]. In those datasets, the names of genes or proteins are labeled as named entities. In each corpus, we randomly took 200 sentences and tested whether the linguistic patterns can be applied. For each sentence, we manually measure the performance of linguistic sequential patterns to detect those interactions. Note that we also carried out a POS tagging of those sentences in order to correctly apply the pattern language. Table 1 presents the scores of the application of the patterns as linguistic IE rules: Precision, Recall and *f-score*[5]. The scores are similar in the three corpora. Moreover, the precision is very good and the recall is correct. Further investigations with different values of the parameter k are a promising issue: indeed, higher k is, more specific patterns are.

Note that the scope of the extracted linguistic IE rules in the experiments is the whole sentence. That scope may introduce ambiguities in the detection of interactions and thus false positives when more than two genes appear in

[5] The used *f-score* function is : $f\text{-}score = \dfrac{2 \times Precision \times Recall}{Precision + Recall}$.

Table 1. Tests on several corpora

Corpus	Precision	Recall	F-Score
BioCreative [21]	0.92	0.767	0.836
GeneTag [20]	0.909	0.8	0.851
AIMed [2]	0.93	0.84	0.88

the same sentence. Several cases are possible: when several binary interactions are present in the sentence, when the interaction is n-ary ($n \geq 3$) or when an interaction is found with a list of genes. The case of n-ary interactions can be solved with a training dataset containing n-ary interactions. The other two cases can be treated by introducing limitations of pattern scope, for example cue-phrases (e.g., *but*, *however*). False negatives mainly depend on the absence of some nouns or verbs of interaction in the patterns. For example, the noun "modulation" is not learned in a pattern whereas the verb "modulate" appears. This suggests that the use of linguistic resources (e.g., lexicon or dictionary) can, manually or semi-automatically, improve patterns and thus interaction detection.

5 Conclusion and Future Work

This paper proposes a method based on a cross-fertilization of sequence mining under constraints and recursive pattern mining to produce a user-manageable set of linguistic information extraction rules, such as the discovery relations between named entities. The constraints enable to drive the mining process towards the user objectives by filtering irrelevant patterns. The recursive sequence mining allows the effective mastering of the number of discovered patterns that are returned. In addition, we prove important properties on recursive pattern mining on sequences. To the best of our knowledge, it is the first approach combining sequential pattern mining methods, constraints and linguistic information.

The case study shows the feasibility and the interest of our method to discover the named entity relations. We have conducted experiments on biomedical textual data to detect gene interactions. Our proposed approach does not need syntactic parsing neither resource except the training data. In addition, the patterns as linguistic IE rules are understandable by a user. From a qualitative point of view, it is interesting to note that the subcategorization of the verbs given by the POS tagging indicates the passive or active verbs and identifies the direction of the relation. Prepositions can also convey that kind of information, which is precious when the pattern does not contain a verb. Promising future work consists of designing more complex linguistic constraints and pushing them within the mining process.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE. IEEE, Los Alamitos (1995)
2. Bunescu, R.C., Mooney, R.J.: A shortest path dependency kernel for relation extraction. In: HLT/EMNLP, pp. 724–731. ACL (2005)

3. Cellier, P., Charnois, T., Plantevit, M.: Sequential patterns to discover and characterise biological relations. In: Gelbukh, A. (ed.) CICLing 2010. LNCS, vol. 6008, pp. 537–548. Springer, Heidelberg (2010)
4. Crémilleux, B., Soulet, A., Klema, J., Hébert, C., Gandrillon, O.: Discovering knowledge from local patterns in sage data. In: Data Mining and Medical Knowledge Management: Cases and Applications, pp. 251–267. IGI Publishing (2009)
5. Fundel, K., Küffner, R., Zimmer, R.: Relex - Relation extraction using dependency parse trees. Bioinformatics 23(3), 365–371 (2007)
6. Garofalakis, M.N., Rastogi, R., Shim, K.: Spirit: Sequential pattern mining with regular expression constraints. In: Proc. Int. Conf. on Very Large Data Bases, pp. 223–234. Morgan Kaufmann, San Francisco (1999)
7. Giuliano, C., Lavelli, A., Romano, L.: Exploiting shallow linguistic information for relation extraction from biomedical literature. In: EACL, pp. 401–408 (2006)
8. Hakenberg, J., Plake, C., Royer, L., Strobelt, H., Leser, U., Schroeder, M.: Gene mention normalization and interaction extraction with context models and sentence motifs. Genome biology 9(Suppl. 2), S14 (2008)
9. Joshi, S., Ramakrishnan, G., Balakrishnan, S., Srinivasan, A.: Information extraction using non-consecutive word sequences. In: Workshop on Text Mining and Link Analysis IJCAI (2007)
10. Krallinger, M., Leitner, F., Rodriguez-Penagos, C., Valencia, A.: Overview of the protein-protein interaction annotation extraction task of BioCreative II. Genome Biology 9(Suppl. 2), S4 (2008)
11. Nanni, M., Rigotti, C.: Extracting trees of quantitative serial episodes. In: Džeroski, S., Struyf, J. (eds.) KDID 2006. LNCS, vol. 4747, pp. 170–188. Springer, Heidelberg (2007)
12. Nédellec, C.: Machine learning for information extraction in genomics - state of the art and perspectives. In: Studies in Fuzziness and Soft Comp. Sirmakessis (2004)
13. Ng, R.T., Lakshmanan, L.V.S., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained association rules. In: ACM SIGMOD (1998)
14. Pei, J., Han, J., Lakshmanan, L.V.S.: Mining frequent itemsets with convertible constraints. In: ICDE, pp. 433–442. IEE Computer Society (2001)
15. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns by prefix-projected growth. In: ICDE, pp. 215–224. IEEE Computer Society, Los Alamitos (2001)
16. Rosario, B., Hearst, M.A.: Multi-way relation classification: Application to protein-protein interactions. In: HLT/EMNLP, pp. 732–739. ACL (2005)
17. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Proc. of Int. Conf. on New Methods in Language Processing (September 1994)
18. Schneider, G., Kaljurand, K., Rinaldi, F.: Detecting protein-protein interactions in biomedical texts using a parser and linguistic resources. In: Gelbukh, A. (ed.) CICLing 2009. LNCS, vol. 5449, pp. 406–417. Springer, Heidelberg (2009)
19. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
20. Tanabe, L., Xie, N., Thom, L.H., Matten, W., Wilbur, J.: GENETAG: a tagged corpus for gene/protein named entity recognition. BMC Bioinformatics 6, 10 (2005)
21. Yeh, A., Morgan, A., Colosimo, M., Hirschman, L.: BioCreAtIvE Task 1A: Gene mention finding evaluation. BMC Bioinformatics 6(Suppl. 1), S2 (2005)
22. Zaki, M.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning 42(1/2), 31–60 (2001)
23. Zweigenbaum, P., Demner-Fushman, D., Yu, H., Cohen, K.B.: Frontiers of biomedical text mining: current progress. Brief. Bioinform. 8(5), 358–375 (2007)