Exploring Pattern Structures of Syntactic Trees for Relation Extraction

Artuur Leeuwenberg, Aleksey Buzmakov, Yannick Toussaint, Amedeo Napoli

Exploring Pattern Structures of Syntactic Trees for Relation Extraction*

Artuur Leeuwenberg, Aleksey Buzmakov, Yannick Toussaint, and Amedeo Napoli

LORIA (CNRS – INRIA – Université de Lorraine)

Email:
t.leeuwenberg@gmail.com,firstname.lastname@loria.fr

Abstract

In this paper we explore the possibility of defining an original pattern structure for managing syntactic trees. More precisely, we are interested in the extraction of relations such as drug-drug interactions (DDIs) in medical texts where sentences are represented as syntactic trees. In this specific pattern structure, called STPS, the similarity operator is based on rooted tree intersection. Moreover, we introduce "Lazy Pattern Structure Classification" (LPSC), which is a symbolic method able to extract and classify DDI sentences w.r.t. STPS. To decrease computation time, a projection and a set of tree-simplification operations are proposed. We evaluated the method by means of a 10-fold cross validation on the corpus of the DDI extraction challenge 2011, and we obtained very encouraging results that are reported at the end of the paper.
KEYWORDS: pattern structures; relation extraction; formal concept analysis; DDI extraction

1 Introduction

When a doctor wants to prescribe a drug to a patient, he/she would like to know when this drug interacts with other drugs that the patient may already take. A lot of research has been done on each drug, resulting in a lot of articles (often more than 1000 articles per drug). It would not be feasible for a human agent to read all these articles. For this reason it could be interesting to automatically find which drugs are interacting in these articles. Accordingly, in the extraction of drug-drug interactions –DDIs in the following– the task is to find pairs of drugs that are described as interacting in a sentence or a text.

*The final publication is available at link.springer.com

In 2011, for the first time, a challenge on this task was initiated [12]. Several methods were proposed to perform this task [14, 3, 4, 2, 11, 7]. The best performing system, i.e. the system with the highest F_1-measure on the given test set, combined several different subsystems in which information from different feature spaces was exploited [14]. Their highest F_1 on the test set was 65.7, and their F_1 for a document-wise 10-fold cross validation on the training data was 60.6. Linguistics features were used, such as part-of-speech, together with different tree kernels of the dependency parses, i.e. trees describing the grammatical dependencies between words, of the sentences. Another system that was successful in the challenge was based on a union of two different machine learning techniques [3]. The first machine learning technique is a feature-based SVM using different words, morphosyntactic features (internal structural features of words, like number and case) and contextual features (words in between the two considered drugs). The second machine learning technique is a kernel-based method combining three different kernels, namely "shallow linguistic information" (like part-of-speech and word-inflection information), "mildly extended dependency trees" and "phrase structure".

It appears that the most successful systems combine both deep linguistic information, such as dependency trees or phrase structures, and shallow linguistic features, such as word features and morphological information. Thus, we propose to apply a symbolic method, based on pattern structures [6] and Formal Concept Analysis (FCA), to deal with the phrase structure, i.e. the syntactic level, in a different way. A pattern structure can manage a complex data type, such as a tree, and allows one to build a hierarchy of elements of this data type, in the present case a hierarchy of trees. Such a pattern structure comes with a classification technique, called "Lazy Pattern Structure Classification" (LPSC) [9], which classifies the syntactic trees containing drug-drug interactions. This is one original application of pattern structures to syntactic trees and to the task of text-mining (here the mining of DDIs). The method is novel and deserves more research work but we already obtained substantial results showing that the current approach is suitable and valuable.

The rest of the paper is organized as follows. Firstly we explain the pipeline on which relies the proposed system. Then we define the pattern structure for syntactic trees, namely STPS, and as well Lazy Pattern Structure Classification (LPSC). After that, we introduce a projection related to STPS and a set of tree-simplification operations to reduce computational time. Finally we evaluate the method on the corpus of the DDI challenge 2011 [12].

2 The Data and the Pipeline

Our data consists of medical texts containing potential drug-drug interactions, i.e. the training corpus of the DDI extraction challenge 2011 [12]. This corpus consists of around 4200 sentences containing around 23000 potential interactions of which a small portion ($\sim$10%) is annotated as positive and the rest as negative. In these data drugs in the sentences are already tagged (see Example 1).

If we take a sentence from the data containing n drugs, there are $\binom{n}{2}$ pairs of drugs in the sentence that can potentially interact. Each such pair is represented by a separate sentence, where the two potentially interacting drugs in the sentence are replaced with a `drug_tag_r` tag, and all other drugs by a `drug_tag` tag (see Examples 2 and 3 where the corresponding tags are following the name of the tagged drug).

Example 1. *Antihistamines (`drug`) may enhance the effects of tricyclic_ antidepressants (`drug`), barbiturates (`drug`), alcohol (`drug`), and other CNS_ depressants (`drug`).*

Example 2. *`drug_tag_r` may enhance the effects of `drug_tag`, `drug_tag`, `drug_tag`, and other `drug_tag_r`.*

Example 3. *`drug_tag` may enhance the effects of `drug_tag`, `drug_tag_r`, `drug_tag`, and other `drug_tag_r`.*

Each such tagged sentence, representing a possible drug-drug interaction, is parsed by the Stanford constituency parser v3.4 [13, 8]. The resulting trees are simplified by means of operations that preserve the parts of the tree describing the potential interaction as much as possible. Trees, representing drug-drug pairs, can be considered as positive or negative. Trees are "positive" when an interaction is described between the two drugs replaced by the `drug_tag_r` tag (Example 2). Trees are "negative" when no such interaction is present (Example 3). The positive simplified tree of Example 2 is shown in Figure 1.

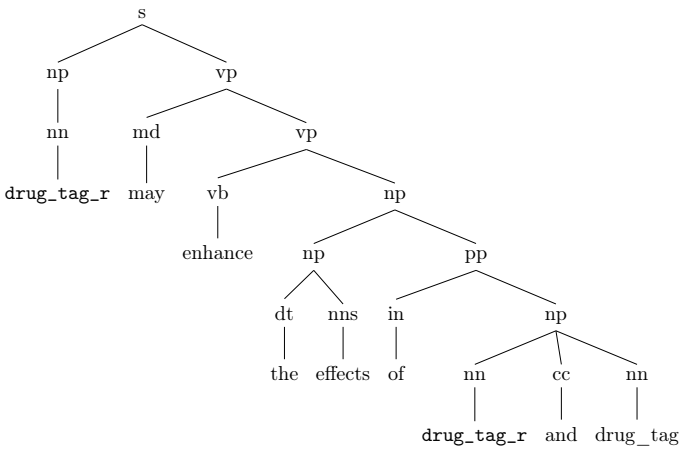

Figure 1: The simplified syntax tree from Example 2.

A pattern structure is defined on such syntactic trees, whose similarity operator is based on unordered rooted tree intersection. The trees are interpreted as unordered w.r.t the constituent order in the sentence in order to be able to generalize over some grammatical structures (eg. conjunctions or enumerations) without losing important grammatical relations (eg. verb argument relations,

prepositions) as they are also encoded in the hierarchy of the tree. To improve the computational time of similarity, a projection is introduced, which can be considered as a simplification of the similarity operator. Pattern structures, similarity and projections are introduced and discussed here after.

The set of trees, obtained from parsing the tagged sentences, is split into a "training set" and a "testing set" and LPSC is used to classify the trees. In the experiments, different settings based on tree simplifications are evaluated. Finally, a schematic view of the pipeline, starting from DDIs and going to LPSC classification, is shown in Figure 2.

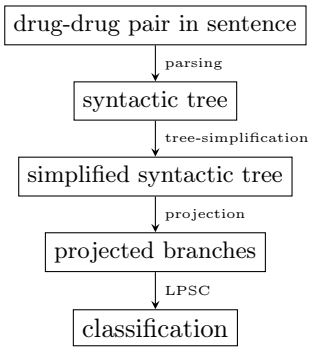

Figure 2: A schematic view of the pipeline.

3 A Pattern Structure for Syntactic Trees

Pattern Structures were introduced in [6] and are a generalization of Formal Concept Analysis (FCA) [15]. From the pattern structure perspective, data can be thought of as a set of objects (G) with corresponding records. Each record is an object description, also called a *pattern*, in contrast to a set of binary attributes, as in standard FCA. On the set of potential descriptions (D) a similarity operator should be defined, which should be idempotent, associative and commutative. In this way, the partial ordering on object descriptions is a semilattice and can be used in a similar way as in Formal Concept Analysis to extract (meaningful) concepts from an unstructured data set, in an unsupervised way. More precisely, we have the following definitions.

Definition 1. *Let G be a set of objects, let $(D, \sqcap)$ be a meet-semilattice of potential object descriptions, and let $\delta : G \to D$ be a mapping. Then $(G, (D, \sqcap), \delta)$ is called a* pattern structure*, provided that the set*

$$\delta(G) := \{\delta(g) | g \in G\}$$

generates a complete sub-semilattice $(D_\delta, \sqcap)$ of $(D, \sqcap)$, i.e. every subset X of $\delta(G)$ has an infimum $\sqcap X$ in $(D, \sqcap)$ and D_δ is the set of these infima [6].

On a pattern structure $(G, (D, \sqcap), \delta)$ a Galois connection can be defined, linking sets of objects with set of descriptions.

$$A^{\diamond} := \prod_{g \in A} \delta(g) \quad \text{for } A \subseteq G$$

$$d^{\diamond} := \{g \in G | d \sqsubseteq \delta(g)\} \quad \text{for } d \in D$$

A concept (A, d) in $(G, (D, \sqcap), \delta)$ verifies $A^{\diamond} = d$ and $d^{\diamond} = A$ where A is the extent and d the intent of the concept. The subsumption order ($\sqsubseteq$) between descriptions c and d is defined as follows:

$$c \sqsubseteq d \Leftrightarrow c \sqcap d = c$$

Concepts are maximal closed sets of objects and their corresponding descriptions. Formal concepts form a concept lattice where the ordering between concepts is given as usual by inclusion of concept extents. The meaning of such a lattice depends on the similarity operator. The same data can be associated with different pattern structures. We look further into defining our pattern structure on trees, in particular syntax trees based on natural language sentences.

3.1 Objects and Object Descriptions

In the current case, the set of objects G in the considered pattern structure $(G, (D, \sqcap), \delta)$ consists of drug-drug pairs, i.e. DDIs, extracted from the collection of sentences. Then the set of object descriptions D is composed of "unordered labeled trees". The resulting pattern structure will be called "Syntactic Tree Pattern Structure" or STPS for short.

Definition 2. *An* unordered labeled rooted tree t *is a simple connected graph* $t = \langle N, E \rangle$*, where* N *is a set of nodes, and* E *a set of ordered pairs from* $N \times N$*, called edges. It should satisfy two conditions:*

- t *does not contain any cycle (it is a tree)*
- t *has one distinguished node* $r \in N$*, called the root node, that is an ancestor of every node* $n \in N$*.*

In unordered labeled rooted trees, nodes carry a label while there exists no order between the children of each node. This means that the trees in Figure 3 are considered to be equivalent.

The mapping δ gives for each potential drug-drug pair the corresponding unordered syntactic tree of the sentence in which it occurs, where the drugs are replaced by the tags. Intuitively, one could think of δ as the function that parses the sentence and simplifies the resulting tree.

Figure 3: Two equivalent unordered labeled rooted trees.

3.2 Similarity Operators

A similarity operator $\sqcap_t$ is defined on the set of object descriptions D. This operator is based on rooted tree intersection. In [1], rooted tree intersection is defined for unordered unlabeled trees and a corresponding algorithm is given. Our definition and implementation follow those in [1], except that we consider trees with labeled nodes.

To define our rooted tree intersection for unordered labeled trees we need to define the notion of rooted subtree first.

Definition 3. *Rooted tree* $t_1 = \langle N_1, E_1 \rangle$ *is a* rooted subtree *of rooted tree* $t_2 = \langle N_2, E_2 \rangle$ *(from now written as* $t_1 \subseteq_t t_2$*) iff the following conditions hold:*

- $N_1 \subseteq N_2$
- $E_1 \subseteq E_2$
- t_1 *and* t_2 *have the same root.*

Using this notion of subtree, we can define a rooted intersection operator on trees.

Definition 4. *The* rooted tree intersection *between tree* t_1 *and* t_2*, from now written as* $t_1 \cap_t t_2$*, is the set containing all maximal trees*[1] *from*

$$\{t \mid t \subseteq_t t_1\} \cap \{t \mid t \subseteq_t t_2\}$$

i.e. the intersection between all subtrees of t_1 *and all subtrees of* t_2*.*

An example of such intersection is shown in Figure 4.

With the notion of rooted tree intersection we can define the similarity operator of our pattern structure.

Definition 5. *The* similarity *between a set of trees* A *and a set of trees* B*, written as* $A \sqcap_t B$*, is the subset of maximal trees from*

$$\bigcup_{(a,b) \in A \times B} a \cap_t b$$

The corresponding subsumption operator is defined as mentioned previously.

$$A \sqsubseteq_t B \Leftrightarrow A \sqcap_t B = A$$

[1]The maximal trees from a set X are all trees of X that are not a rooted subtree of another tree in X.

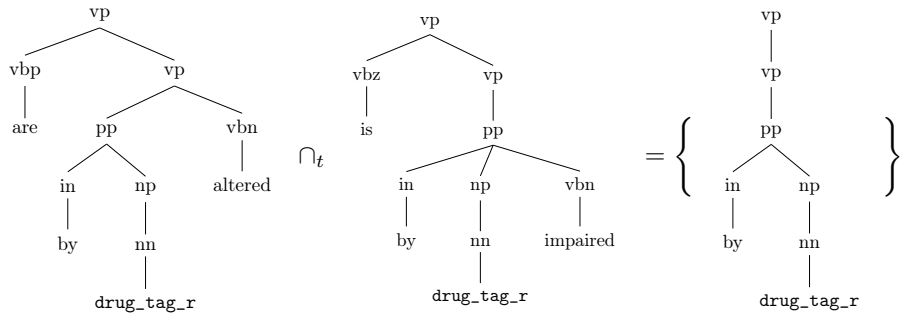

Figure 4: An example of rooted unordered tree intersection ($\cap_t$) of two syntactic tree fragments. The tree on the right side is the maximal rooted subtree of both trees on the left side.

3.3 The Projections for the Syntactic Tree Pattern Structure

Projections for pattern structures were introduced in [6]. A projection is used for weakening the object descriptions and for allowing better performances in computation. Moreover, "good" projections always try to minimize the loss of information.

Definition 6. *A projection of a pattern structure $(G, (D, \sqcap), \delta)$ is a mapping $\psi : D \to D$ that replaces every object description $d \in D$ by $\psi(d)$, such that the original pattern structure is replaced by $(G, (D, \sqcap), \psi \circ \delta)$. It is required that ψ is a kernel operation, i.e. ψ is*

monotone: if $x \sqsubseteq y$, then $\psi(x) \sqsubseteq \psi(y)$,

contractive: $\psi(x) \sqsubseteq x$, and

idempotent: $\psi(\psi(x)) = \psi(x)$.

Projections can be used efficiently to reduce computation time of the similarity operator. For pattern structures of graphs several projections were already proposed and applied in the chemical domain [10]. Here we propose a projection that maps each tree description onto the set of its maximal branches.

Definition 7. *Rooted tree $t_1 = \langle N_1, E_1 \rangle$ is a* branch *of rooted tree $t_2 = \langle N_2, E_2 \rangle$ iff the following conditions hold:*

- $t_1 \subseteq_t t_2$
- *Each node $n_1 \in N_1$ has at most one outgoing edge.*

Definition 8. *The branch projection of a set of rooted trees T, from now written as $\psi_b(T)$, is the set of maximal trees from*

$$\bigcup_{t \in T} \{b \mid b \text{ is a branch of } t\}$$

Thus, a tree t defined by a root with n leaves will be projected to a set of size n, containing its branches (see Figure 5).

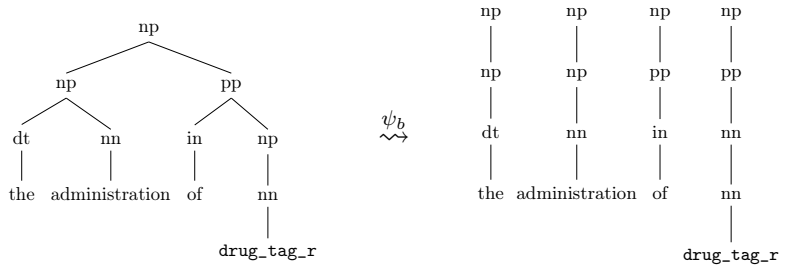

Figure 5: A tree and its maximal branches.

4 Classification based on Lazy Hypothesis Evaluation

Actually, the concept lattice resulting from the pattern structure which is defined above has not to be built. Instead, we follow a (kind of) supervised classification method for determining objects whose description includes a syntactic tree effectively representing a DDI, i.e. the drugs lying in the syntactic tree and marked with `drug_tag_r` tags are interacting. We follow a "Lazy Pattern Structure Classification" (LPSC) introduced in [9]. LPSC can classify objects from a given pattern structure in polynomial time w.r.t the cardinality of the set of objects G considered as training data. It is based on a set of positive examples G_+ and a set of negative examples G_-. In the current experiment, positive examples are sentences including interacting drug-drug pairs while negative examples are sentences which do not include interacting drug-drug pairs.

In [9], the classification of a new object o_n is performed w.r.t. two questions:

(1.) Is there a "positive hypothesis" for o_n?

(2.) Is there a "negative hypothesis" for o_n?

A positive hypothesis is defined as a pattern intent in the pattern structure $(G_+, (D, \sqcap), \delta)$ that does not subsume any pattern from $\delta(G_-)$, i.e. does not subsume any negative example. A positive hypothesis for o_n is found iff:

$$\exists g_+ \in G_+ \ \forall g_- \in G_- : (o_n \sqcap g_+^\diamond) \not\sqsubseteq g_-^\diamond$$

1.	2.	Classification
yes	yes	undefined
yes	no	positive
no	yes	negative
no	no	undefined

1.	2.	Classification
yes	yes	positive
yes	no	positive
no	yes	negative
no	no	negative

Table 1: Criteria in Lazy Pattern Structure Classification according to [9] are displayed on the left, and criteria in LPSC restricted to only positive hypotheses evaluation –used in our experiments– are displayed on the right.

In other words, a positive hypothesis for o_n is found if and only if o_n is similar to a positive example g_+, i.e. the potential positive hypothesis, and o_n does not share this similarity with any negative example g_-. A negative hypothesis for o_n is defined symmetrically, by switching the negative and positive examples. How an object is classified depends on the answers for the questions **(1.)** and **(2.)**, as shown in Table 1.

Our classification criteria differ from that in [9] as we are only looking for positive hypotheses and not for negative hypotheses. The underlying idea is that we assume that typical syntactic trees containing a DDI have some characteristic structures, while trees that do not contain any DDI do not have such characteristic structures. Thus, we discriminate positive and negative hypotheses w.r.t. the classification criteria, contrasting with [9] where there are also unclassified objects. In our experiment, an object is classified as positive when the first question is answered with "yes", and by complementarity, an object is classified as negative when this first question is answered with "no". This kind of classification was exclusively used in our experiments and is termed as "*Lazy Positive Hypothesis Classification*" (LPHC) (see Table 1).

An example of a positive hypothesis that was found in the experiments with LPHC is shown in Figure 6. This positive hypothesis was created when classifying the tree corresponding to the potential DDI described in Example 4. Moreover, the positive example from the training set is the tree corresponding to Example 5.

Example 4. *Antihistamines (`drug_tag_r`) may partially counteract the anticoagulation effects of heparin (`drug_tag_r`) or warfarin (`drug_tag`).*

Example 5. *Tricyclic_ antidepressants (`drug_tag_r`) may block the antihypertensive (`drug_tag`) action of guanethidine (`drug_tag_r`) and similarly acting compounds.*

The tree in Figure 6 materializes the similarity between Example 4 and Example 5, and is not subsumed by any negative example in the training data. For this reason it is classified as a positive hypothesis for Example 4.

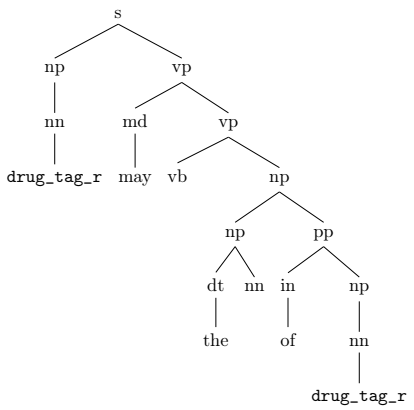

Figure 6: A positive pattern found in the experiments, created from the two sentences in Example 4 and Example 5. It should be noticed that, for the sake of clarity, this pattern is represented as a tree respecting the word ordering, but actually it is an unordered set of branches.

5 The Simplification of Syntactic Trees

When we looked manually at the sentences in the dataset, we remarked that not all parts of some sentences seem to contain useful information about the described DDIs. When a syntactic tree is large, it often takes more time to compute similarity with other trees. Therefore, it is interesting to remove parts of the sentence that are not required to find a DDI. Accordingly, we introduce "tree simplification operations" which are described below.

Constituent simplification.

By means of manually checking the trees, we noticed that some of the constituents are not very informative for describing a DDI in a sentence. In Example 6, it can be seen that an interaction is described between two `drug_tag_r` tags.

Example 6. *In diabetic patients, the metabolic effects of* ***drug_tag_r*** *may decrease blood glucose and therefore* ***drug_tag_r*** *requirements.*

However, it can be seen in Example 7 that some parts of the sentence can be removed without altering the description of the interaction.

Example 7. *The effects of* ***drug_tag_r*** *may decrease blood glucose and* ***drug_tag_r*** *requirements*

Usually, we can remove the constituents when the tree corresponding to the constituent does not contain any of the possibly interacting drugs, i.e.. any of the two `drug_tag_r` nodes.

The candidate constituent to be removed that we considered are: (i) adjectives (JJ), (ii) prepositional phrases (PP), (iii) declarative clauses and clauses introduced by a subordinate conjunction such as relative clauses (S, SBAR), (v) adverbal phrases (ADVP) and (vi) parenthetical expressions (PRN). Subtrees of all these six categories that do not contain any of the `drug_tag_r` nodes are removed from the initial tree. The simplification of the tree corresponding to Example 6 is given in Figure 7.

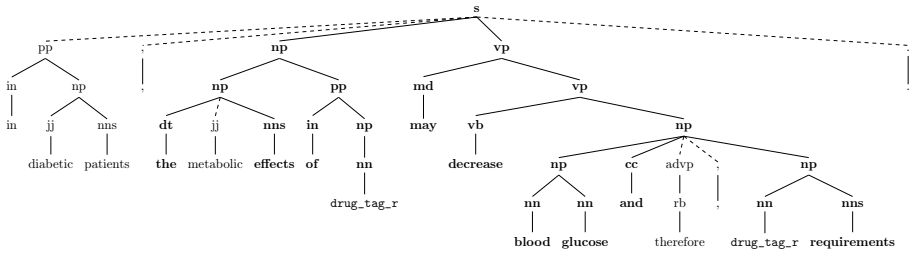

Figure 7: The original syntactic tree associated with the sentence "In diabetic patients, the metabolic effects of `drug_tag_r` may decrease blood glucose and therefore `drug_tag_r` requirements." The subtrees that will fall off after simplification are indicated with dashed lines.

NEGVP renaming.

To deal on a simple level with negation, each VP-node, i.e. representing a verb phrase, that directly contains a negating expression (not/no) is renamed as a NEGVP node. In this way a normal VP will not be matched with, or considered similar to, a negated VP.

Lowest-S simplification.

Because relations can sometimes be described very deep in a subordinate clause, only the deepest S-node (i.e. declarative clause) containing both `drug_tag_r` tags is considered, as shown in Figure 5. This makes sure that deeply nested interaction descriptions can be compared in an easier way to surface interaction descriptions. This way the lowest-S constituent in Example 8 (i.e. in the inner brackets) can be compared to the sentence in Example 9.

Example 8. *[$_S$ `drug_tag`: Clinical studies, as well as post marketing observations, have shown that [$_S$ `drug_tag_r` can reduce the `drug_tag` effect of `drug_tag_r` and `drug_tag` in some patients].]*

Example 9. *[$_S$ `drug_tag_r` agents reduce the renal clearance of `drug_tag_r` and add a high risk of `drug_tag` toxicity.]*

However, this rule does not always preserve all crucial information about the potential DDI. In some cases important information can be described at a meta level.

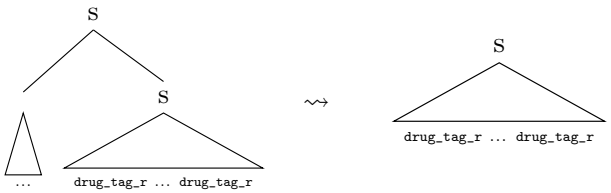

Figure 8: Schematic view of lowest-S simplification.

Example 10. *[S It is not known if [S* ***drug_tag_r*** *differ in their effectiveness when used with* ***drug_tag_r****].]*

In Example 10, both `drug_tag_r` tags occur in the S-constituent indicated by the inner brackets. Thus, when using lowest-S simplification, only the expression in the inner brackets is considered. However, the expression outside of the brackets, i.e. "It is not known if..." contains important information about the DDI description inside. It weakens or even nullifies the interaction that is described inside. For now, we do not have any clear solution to deal with such cases and we ignored them.

Link contraction.

After applying the constituent simplification operation, a resulting tree might contain branches that link nodes holding the same label with only one child. Such cases can be considered as redundant and can be simplified by removing the redundant non-branching duplicate nodes and linking the contracted new node with its single child node. An example is given in Figure 9.

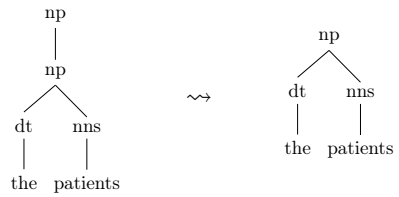

Figure 9: A tree and its contracted version.

If we apply all these tree simplifications on the trees obtained after parsing the experiment dataset, the average number of nodes in each tree drops from 130 to 41 and the maximum number of nodes from 311 to 138. This shows that the application of these simplification operations have a substantial impact on the set of resulting syntactic trees.

6 Experiments and Discussion

6.1 The Experiment

In this experiment, different settings were evaluated. Each system classifies the potential DDIs by means of lazy pattern structure classification (actually positive hypothesis classification or LPHC). The underlying pattern structure is the one which is described in Section 3, using the branch projection. The settings are differing only in the tree simplifications that were applied.

For each setting, a 10-fold cross validation was performed on the data set. The corpus that is used is the training corpus of the DDI extraction challenge 2011 [12].In this corpus, the drugs are annotated and the interactions are build using the DrugBank, and then manually checked by a domain specialist.

We ran the experiments on a laptop with an i7 Intel processor (using 4 of its 8 virtual cores). The algorithm was implemented in Python. On average, each object classification took around 2 seconds. This long duration is primarily due to the search for positive hypotheses for each classification. It could be also possible to extract these positive hypotheses on a training set offline. Then they could be used as features in a different classification paradigm, maybe more optimized for a particular task. Here we did not do this as we were mostly interested in increasing the quality of the patterns.

Simplifications	**P**	**R**	**F_1**
1. NEGVP, lowest-S, contraction	0.29786	0.48900	0.37022
2. NEGVP, contraction	0.32261	0.39044	0.35330
3. lowest-S, contraction	0.27073	**0.49450**	0.34990
4. NEGVP, lowest-S	0.33598	0.44712	0.38367
5. NEGVP, lowest-S, vp-map	0.35216	0.44585	0.39350
6. NEGVP, lowest-S, vp-map, prep-map	**0.38556**	0.41328	**0.39894**

Table 2: Results from 10-fold cross validation on the DDI 2011 data set. Performance is measured in precision (P), recall (R) and F_1-measure (F_1). In all conditions constituent simplification is applied.

The results from six settings we tested in the experiment are shown in Table 2. When we look at condition 1 and 2 in Table 2, we can see that applying lowest-S simplification strongly increases the recall, by 9.9%, but also reduces precision by 2.5%. Overall, F_1 increased by 1.69%. The reduction in precision, is probably due to some cases where the interaction is not fully described in the lowest declarative clause (lowest S-node). The increase in recall is probably due to the fact that surface clauses can now be compared better to deeper ones.

Applying the link contraction seems to have a weaker but similar effect. However, it decreases the F_1-measure. This can be noticed if we compare setting 1 and 4. After applying link contraction, the precision reduces with 3.8%, while the recall increases with 4,2% and the F_1-measure decreases with 1.4%. It

appears that even if trees are non-branching, the hierarchy and its depth are important. Furthermore, if we compare setting 1 and 3, we can see that the NEGVP renaming has a positive effect on precision and only a minor negative effect on recall. It increases the F_1-measure with 2%.

Settings 5 and 6 are discussed below, in the error analysis.

6.2 Error Analysis

We manually looked both at false positives (i.e. negative trees classified as positive) and false negatives (i.e. positive trees classified as negative). False positives can be analyzed very precisely, because for each positively classified tree, the positive hypothesis from the positive training examples can be examined as well. A few non-mutually exclusive error categories that we found are the following.

1. *Insufficient similarity:* Sometimes, the similarity between the to be classified drug-drug pair and the positive hypothesis is too small to make a proper classification. This can be due to data sparseness or lack of information in the trees. Often in these cases the similarity between the to be classified tree and its corresponding positive hypothesis does not even contain a verb phrase node. Another frequent case is that the prepositions in the to be classified tree and its positive hypothesis do not match. An example of such poor similarity is given in Figure 10.

2. *Non-sentences:* Some mistakes seem to occur in phrases that are not full sentences or that are not parsed as such. Often the parser considers these phrases as noun phrases or as "fragments" (i.e. the root node is NP or FRAG). A reason for errors to occur in this category can be that there is not enough training data for these cases, or the parser made a mistake. Again the pattern in Figure 10 is an example of a non-sentence (an NP).

3. *Mistakes in annotation:* In some cases, a misclassification is due to errors in the drug annotations or in the interaction annotations. Examples of such cases can be found in [14].

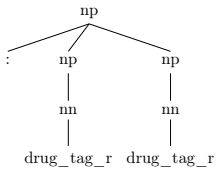

Figure 10: An example for error category 1 and 2. This pattern is clearly not sufficient for classification. This is due to the lack of a negative example in the training data that subsumes this pattern.

It can be noticed that some patterns may cause false positives, but can at the same time be responsible for a lot of true positives. In our experiments, we

did not do any filtering directly based on performance. When the interest is in pure performance, it could be interesting to filter patterns that do not cause any true positives or those that cause more false positives then true positives.

6.3 Similarity Mappings

In error category 1, the similarity between the to be classified tree and its positive hypothesis was too small to make a proper classification. To prevent insufficient similarity, one could manually introduce some linguistically based constraints on the hypotheses and exclude hypotheses that do not satisfy them. We do this by mapping outputs of the similarity operator that do not fulfill the constraints to the empty set, and therefore have no potential for being a hypotheses. Based on the found errors, we introduce two types of similarity mappings: (i) *VP-mapping*, which maps outputs of the similarity operator that do not contain either a VP-node or a NEGVP-node to the empty set, (ii) *Prep-mapping*, which maps outputs of the similarity operator that do contain a prepositional phrase (PP-node) but not the exact preposition to the empty set.

Their result on performances can be found in Table 2. When we compare settings 4 and 5, the "vp-mapping" seems to have a small positive effect on precision (+ 1.6%), and hardly any effect on recall. When we compare settings 5 and 6, the "prep-mapping" also seems to have a positive effect on precision (+ 3.34%). However, the recall seems to decrease as well (- 3.3%). A reason for this could be that a side effect of the "prep-mapping" is that if two trees share a PP-node, but do not share the same preposition, this is considered the same as no PP-node match at all.

7 Conclusions and Future Work

In this paper we presented a new way of analyzing drug-drug interactions in sentences based on FCA. We defined a pattern structure and introduced a projection for syntactic trees. Lazy pattern structure classification was also used to discover informative syntactic patterns, i.e. including DDIs. Furthermore we introduced a set of tree-simplification operations to reduce the size of the syntactic trees. The whole method was evaluated on the training corpus of the DDI extraction challenge 2011.

At present, it can be concluded that in terms of performance the system in its current state does not achieve very high performance. This is probably due to the rigid way the system deals with the found patterns. Furthermore, it should be noticed that this is a single system, using only phrase structure information.

However, from a qualitative point of view, many extracted syntactic patterns seem quite promising. For example, it would be interesting to use these extracted patterns as features in other classification paradigms and this could be included in future research. Another important direction could be to apply parse thickets [5] for the task of DDI detection. A parse thicket is a graph built

from the set of syntactic trees of a paragraph. This graph is enriched with the semantic links such as pronoun redirections. The work in [5] is based on pattern structures and, hence, can be adapted to our framework. Finally, other possible future research work could include the search for negative hypotheses, and to enrich the syntactic trees with semantic or morphological features.

References

[1] José L Balcázar, Albert Bifet, and Antoni Lozano. Intersection algorithms and a closure operator on unordered trees. *MLG*, page 1, 2006.

[2] Jari Björne, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. Drug-drug interaction extraction from biomedical texts with svm and rls classifiers. *Proceedings of DDIExtraction-2011 challenge task*, pages 35–42, 2011.

[3] Faisal Mahbub Chowdhury, Asma Ben Abacha, Alberto Lavelli, and Pierre Zweigenbaum. Two different machine learning techniques for drug-drug interaction extraction. *Challenge Task on Drug-Drug Interaction Extraction*, pages 19–26, 2011.

[4] Md Faisal Mahbub Chowdhury and Alberto Lavelli. Drug-drug interaction extraction using composite kernels. *Challenge Task on Drug-Drug Interaction Extraction*, pages 27–33, 2011.

[5] BorisA. Galitsky, Dmitry Ilvovsky, SergeiO. Kuznetsov, and Fedor Strok. Finding maximal common sub-parse thickets for multi-sentence search. In Madalina Croitoru, Sebastian Rudolph, Stefan Woltran, and Christophe Gonzales, editors, *Graph Structures for Knowledge Representation and Reasoning*, volume 8323 of *Lecture Notes in Computer Science*, pages 39–57. Springer International Publishing, 2014.

[6] Bernhard Ganter and Sergei O Kuznetsov. Pattern structures and their projections. In *Conceptual Structures: Broadening the Base*, pages 129–142. Springer, 2001.

[7] Sandra Garcia-Blasco, Santiago M Mola-Velasco, Roxana Danger, and Paolo Rosso. Automatic drug-drug interaction detection: A machine learning approach with maximal frequent sequence extraction. *Challenge Task on Drug-Drug Interaction Extraction*, pages 51–58, 2011.

[8] Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[9] Sergei O Kuznetsov. Fitting pattern structures to knowledge discovery in big data. In *Formal Concept Analysis*, pages 254–266. Springer, 2013.

[10] Sergei O Kuznetsov and Mikhail V Samokhin. Learning closed sets of labeled graphs for chemical applications. In *Inductive Logic Programming*, pages 190–208. Springer, 2005.

[11] Anne-Lyse Minard, Lamia Makour, Anne-Laure Ligozat, and Brigitte Grau. Feature selection for drug-drug interaction detection using machine-learning based approaches. *Challenge Task on Drug-Drug Interaction Extraction*, pages 43–50, 2011.

[12] Isabel Segura-Bedmar, Paloma Martınez, and Daniel Sánchez-Cisneros. The 1st DDIExtraction-2011 challenge task: Extraction of drug-drug interactions from biomedical texts. *Challenge Task on Drug-Drug Interaction Extraction*, 2011:1–9, 2011.

[13] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer, 2013.

[14] Philippe Thomas, Mariana Neves, Illés Solt, Domonkos Tikk, and Ulf Leser. Relation extraction for drug-drug interactions using ensemble learning. *Challenge Task on Drug-Drug Interaction Extraction*, pages 11–18, 2011.

[15] Rudolf Wille. *Restructuring lattice theory: an approach based on hierarchies of concepts*. Springer, 2009.