Generalized Framework for Syntax-based Relation Mining

Bonaventura Coppola [†], Alessandro Moschitti [†], Daniele Pighin [†‡]
[†] University of Trento, Department of Computer Science and Engineering
[‡] FBK-Irst, Human Language Technologies Research Unit
{coppola,moschitti}@disi.unitn.it, pighin@fbk.eu

Abstract

Supervised approaches to Data Mining are particularly appealing as they allow for the extraction of complex relations from data objects. In order to facilitate their application in different areas, ranging from protein to protein interaction in bioinformatics to text mining in computational linguistics research, a modular and general mining framework is needed. The major constraint to the generalization process concerns the feature design for the description of relational data.

In this paper, we present a machine learning framework for the automatic mining of relations, where the target objects are structurally organized in a tree. Object types are generalized by means of the use of roles*, whereas the relation properties are described by means of the underlying tree structure. The latter is encoded in the learning algorithm thanks to kernel methods for structured data, which represent structures in terms of their all possible subparts. This approach can be applied to any kind of data disregarding their very nature.*

Experiments with Support Vector Machines on two text mining datasets for relation extraction, i.e. the PropBank and FrameNet corpora, show both that our approach is general, and that it reaches state-of-the-art accuracy.

1. Introduction

Mining relations from text is one of the most interesting Data Mining (DM) problems, testified by several important applications in bioinformatics [2], medicine [28], and other areas such as hypermedia, e.g. for the automatic generation of hyperlinks between related entities across documents [15] or digital media indexing and integration [24]. In bioinformatics, studies on relation mining are carried out on three main different data types: natural language texts, molecular structures expressed in text format (e.g. the DNA sequence), and molecular models as proteins. This research field has also impact in the development of annotated corpora to be employed for the setup of supervised learning frameworks, as in the case of the GENIA corpus [14]. Moreover, there are also trends to widen and generalize the data mining perspective so that multiple structured information sources may be considered at the same time, as in the case of Multi-Relational approaches [6]. Therefore, the design of a framework able to extract relations independently of data is a challenging and interesting research area.

In the specific case of relational mining from texts, two interesting computational linguistics projects, PropBank [21] and FrameNet [1], proposed different approaches for modeling the relations between sentence constituents, i.e. grammatically and semantically meaningful sequences of words. This kind of information is expressed in the form of predicate argument structures (PAS), where a particular word, i.e. the *target*, evokes an action, a situation or an event and establishes a relation among the above constituents.

In particular, given a predicate target word like a verb, a Semantic Role Labeling (SRL) system identifies and properly labels the word sequences that play some *role* with respect to the target word. The roles typically express semantic relations between the target and one of its *arguments*, as in *"John gave Mary the ball"* where *John* is the GIVER in the action expressed by the verb *give*, *the ball* is the GIVEN_OBJECT, and *Mary* is the RECIPIENT of the object.

An interesting aspect is that such relations are derived from the syntactic structure of the referring sentence, that is its syntactic parse tree, where the semantic roles have been annotated. This makes it possible to mine dependencies between words which are located in distant sentence positions. Such goal can be achieved with very high accuracy, as shown by recent SRL works [3, 16].

The abstract function of an SRL system is to mine semantic relations between objects described in a syntactic structure. Therefore, similar learning techniques could be applied to relation extraction in other domains in which a generic syntactic/structural organization for data objects is available. The main problem in generalizing the SRL idea is that the features extracted from the syntactic structures must be able to describe potentially very different objects. For example, in SRL, linguists have designed features like the *head word* or *passive/active sentence form*. Probably, these features do not make any sense in a task of protein classification, where the syntactic structure simply describes the spatial and interaction properties between different proteins' molecules (i.e. the objects specific to that problem).

A viable approach to automatic feature design for structure representation consists in extracting all possible structured features, and then selecting those most significant. This can be achieved by exploiting two major recent findings of the Statistical Learning Theory, i.e. Support Vector Machines (SVMs) [27] and Kernel Methods [25]. The latter can be used to implicitly generate the space of all possible substructures from the target object structure. That is, kernels extract all object properties, while SVMs can emphasize the role of the meaningful substructures, realizing an implicit side-effect of feature selection. In this context, several kernel families (such as the polynomial and the string kernels [25] or the syntactic tree kernels [5]) can be used for the representation of structured objects and their syntactic relations in the task of deriving semantic properties.

Such framework can be further generalized by considering that a domain is typically characterized by local subdomains for which specific *local* relations and roles should be considered. We also informally refer to such subdomains as to *frames*. In other words, an ad-hoc relation miner for any sub-domain can be implemented, where the relations between different sub-domains should be taken into account for its design. Instances of such complex scenario can be found in computational linguistics research, e.g. the FrameNet project, which defines a hierarchical organization of frames according to different frame-to-frame relations. More specifically, each frame clusters together different target words evoking semantically similar predicates associated with the same roles as in the GIVING example above. Additionally, relations between frames, e.g. inheritance and specialization, are defined.

How to effectively and efficiently organize this complex extraction framework is an interesting subject, as it involves the design of innovative data mining algorithms both from a machine learning and from an engineering viewpoint.

In this paper, we propose a general framework for the mining of semantic relations between objects structured in a tree-based hierarchy. Such a framework allows for the use of different miners associated with different frames which establish similar relations on homogeneous semantic roles, the latter being a generalization of object instances. In this preliminary study, the relations between different frames are provided as a prior knowledge and they can be exploited to enhance the extraction of the individual relations described by each frame.

The main properties of our frameworks are:

- Supervised learning of semantic relations among objects organized in a tree structure;
- SVM-based relational miners learnt from examples;
- Kernel Methods for the representation of structured data in terms of trees and sequences;
- Different miners associated with specific local clusters of objects;
- Modular and efficient models, where the detection of a relation and its classification are implemented in multiple steps;
- A joint model to take into account the interdependencies of the mining objects, and able to considers global properties by enforcing different kinds of prior knowledge.

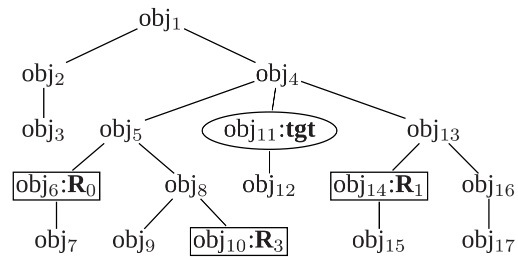


Figure 1. Abstract structure encoding objects and relations.

To test the characteristics of our framework, we experimented with linguistic domains since two very large and popular datasets are available, produced by the FrameNet and the PropBank project. The results on both corpora show state-of-the-art accuracy.

In the remainder of this paper, Section 2 presents a general framework for the extraction of relational patterns from structured input data; Section 2.3 extends the model so that it can be applied to a multi-domain scenario; Section 3 discusses how kernel methods can be applied to discover novel and relevant features in the presence of highly structured data; Section 4 reviews the previous work in this field; Section 5 details the setup and outcome of our experiments; finally, Section 6 summarizes the discussion and presents our conclusions.

2. Mining relational patterns among structurally organized objects

The main task of data mining is to discover interesting patterns from data in a target domain, where data are often structurally organized. Structure is an important source of information that helps in discovering the relationships between different data objects. For example, Figure 1 shows a set of elements organized into a tree structure, where object-to-object relationships are likely to be characterized both by object properties and by the surrounding structure. In particular, we assume that some elements, the *targets* (*tgt*), trigger the relations between other objects, i.e. their *arguments*. The former will also be referred to as *predicates*, whereas the classes of argument objects (those involved in the relations) are called *roles* (e.g. R_0, R_1, R_2).

Previous work on relation mining mainly concerns with unsupervised approaches [7]. However, since such models are based on frequency counts, they are not accurate enough to mine specific and infrequent relationships. In this perspective, supervised approaches are an interesting alternative if we already know the roles and the classes of predicates that we may expect to find in a domain.

A supervised approach assumes that examples of the target objects are available, and that their detection can be automatically learnt. A more interesting step relates to mining relations between such objects as relational patterns. In this section, we describe a general algorithm that, given a predicate, selects its arguments by also classifying their role. A joint model that considers the relations among multiple objects is presented, along with an example on predicate argument extraction from texts in a linguistic domain.

2.1 Object selection and classification

Suppose that our objects are structurally encoded into trees, as shown in Figure 1. The process of recognizing relational patterns, i.e. predicate argument structures, can be intuitively decomposed into two smaller problems:

1. given the predicate object *tgt* (the one triggering the relation), the elements (e.g. the nodes of an XML document) that participate in some relation with such predicate must be identified. We define this task *node selection* (NS) or *argument selection*.
2. The most appropriate role must be assigned to each of the previously selected nodes. We define this task *role classification* (RC).

Considering the example in Figure 1, given the target node obj_{11} (labeled as *tgt*), NS selects its argument nodes (obj_6, obj_{10} and obj_{14}). Then, the appropriate role labels, (R_0, R_3 and R_1 respectively) must be assigned by RC.

Both NS and RC are typically and individually modeled as a supervised learning problem: several classifiers are trained on a dataset of previously labeled data, where the correct annotation is provided for each input instance and target element therein.

A detailed analysis of such approach is shown in Figure 2 and it is hereby described.

First, to improve the efficiency of the classification steps, we enforce any available prior knowledge regarding nodes

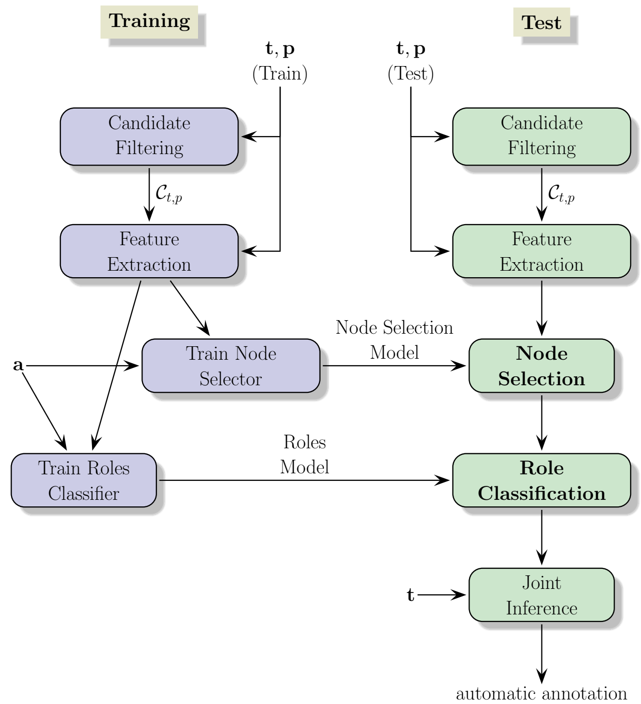


Figure 2. Relational Mining Architecture (RMA).

that do *not* participate in any relation. Thus, we apply a pre-processing stage of *candidate filtering* to both training and test sets. For example, in the case of XML data, we may discard all the comment elements and all those classes of elements that contain indexing information or metadata. The result of this pre-processing is a set $\mathcal{C}_{t,p}$ of all the candidate elements of the tree t that might be arguments of the predicate p.

Second, according to our classification framework, each element $c \in \mathcal{C}_{t,p}$ should be represented in terms of feature vectors capturing the structural properties linking p and c (feature extraction step).

Third, we split the training candidates into: $\bigcup_{t,p} \mathcal{C}^+_{t,p}$, the set of objects which are arguments of at least one predicate, and $\bigcup_{t,p} \mathcal{C}^-_{t,p}$, the set of non-argument objects, where t is a tree of our dataset. These two sets of positive and negative instances, according to some gold standard annotation a, are used to learn a binary classifier implementing NS.

Next, $\bigcup_{t,p} \mathcal{C}^+_{t,p}$ is divided into as many sets as the number of role types, so that a role multi-class classifier RC can be learnt. Since no information about the other roles involved in a relation is available to NS and RC, a joint inference model can be learnt considering alternative outcomes of the classifiers. The joint inference step can be arbitrarily complex, ranging from label-sequence correction schemes [26] to whole probabilistic frameworks built on top of NS and RC output [12].

Finally, at test time NS, RC, and the joint inference model can be used to classify new data as shown on the right side of Figure 2. This supervised machine learning setting constitutes a Relational Mining Architecture (RMA). Section 3 will focus on its critical aspects of object representation and feature extraction.

2.2 Predicate argument mining from texts

In the case of linguistic applications, data are typically structured into syntactic parse trees, and the following mapping holds with respect to Figure 1. The objects (the tree nodes) that we classify are *syntactic constituents* (sequences of words which constitute grammatically and semantically meaningful text fragments). Target nodes correspond to *predicates*. i.e. specific classes of words (usually verbs) which determine linguistic relations among syntactic constituents. Tree leaves correspond to the actual words of the sentence. Hence, the argument nodes are the syntactic constituents participating to the relation triggered by the predicate.

Figure 3.a shows as an example the syntactic parse tree of the sentence *"John took the book and read its title"*, where the circled nodes are predicates and the boxed constituents are the arguments. Note that the argument *John* is shared by both predicates.

So far, we described the general approach for mining relational patterns in a supervised machine learning setting. The generality of this approach through different data mining applications is mostly given by the feature extraction step, in which the designer should include the prior knowledge about a specific task. While Section 3 shows how kernel methods provide a successful technique to automatize feature design, we now focus on extending the basic RMA to multiple sub-domains.

2.3 Extending the approach to multi-frame scenarios

In the previous sections, we introduced a Relational Mining Architecture (RMA) which operates under the assumption of *globally* defined relations over data, so that all the objects to be classified share the same set of relation types and role labels. However, in more complex problems these properties may be only *locally* shared, thus defining subdomains whose local semantics (relation types and role labels) is often referred to as a *frame* (see Section 5.2 for a further example).

Many data mining problems are inherently multi-frame, i.e. the target dataset is naturally partitioned into subsets which share a local semantics. Multi-frame problems need a more complex architecture. Our approach to deal with them is replicating the basic RMA for each sub-domain. Therefore, given k different frames $F^1, \ldots, F^k$, we instantiate k corresponding RMA modules $M^1, \ldots, M^k$.

In general, the extent to which two frames F^i and F^j are actually separated depends on the specific application domain. In fact, frame-to-frame relations often hold, including: similarity, specialization/generalization, and inheritance. These, along with the possibility of sharing specific role labels and predicates across different domains, allow for the definition of several schemes of interdependent RMAs. As a result, the two corresponding modules M^i and M^j may share the role label set, the training data, or even the learning models. A key strength of our multi-frame architecture is to allow a *selective information sharing* between modules. As a typical case, M^i and M^j can share a common node selection model NS, and keep separated their role classification models RCs, or vice versa. In general, three main learning-and-test modalities are allowed:

Per-frame learning: a separate model is instantiated for each module and for each classification stage, i.e. for node selection (NS) and role classification (RC).

Selective learning: given a partition $\mathcal{P}$ over the set of frames, a different classification model (for either or both NS and RC) is instantiated for each frame subset $s \in \mathcal{P}$.

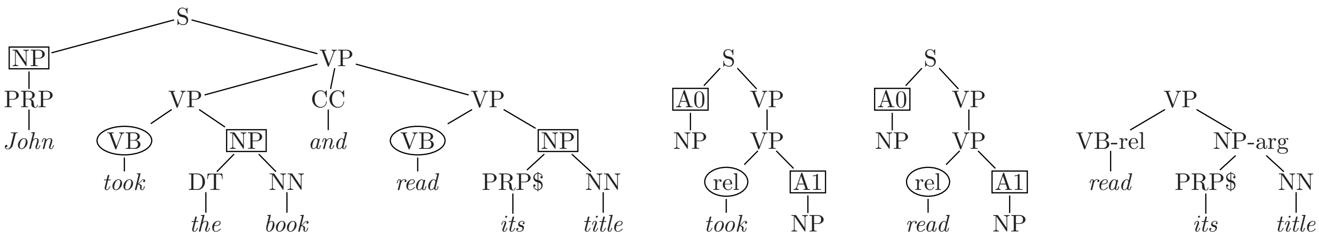
Aggregate learning: all the modules share the same models for NS and RC.

Our implementation of this multi-frame architecture leaves to the user the capability of selecting and customizing the above options through a simple description language.

3. Automatic Structured Feature Generation

Different data mining domains involve different objects and structures, whose individual parts are interesting for detecting the relationships between two objects. The encoding of structured data as feature vectors in a learning algorithm is a complex activity, and it requires remarkable expertise to detect the meaningful subparts. A constructive automatic approach would include all possible substructures as features, and then select the most relevant ones. Since their number is exponential in the number of objects, no machine learning algorithms could manage the resulting feature space dimension.

Support Vector Machines (SVMs) are a very accurate supervised learning approach [27] which allows for the use of kernel functions to evaluate object similarity in very high dimensional and implicit feature spaces. By means of kernel methods, SVMs can carry out learning by using all possible substructures. Moreover, given their robustness to less relevant features, no feature selection step is eventually needed.



(a) A sentence parse tree (b) AST_n for *took* (c) AST_n for *read* (d) AST_1

Figure 3. A syntactic parse tree and subtrees capturing dependencies between roles and predicates.

In the remainder of this section, we describe the kernels for structured data that we use in our framework. We currently limit the object structures to trees, but kernels for more general graphs are also available in literature.

3.1 SVMs and the kernel trick

Supervised learning is based on the use of labeled examples, generally described by means of feature vectors in a n-dimensional space over real numbers, $\Re^n$. Support Vector Machines define a hyperplane $H(\vec{x}) = \vec{w}\cdot\vec{x}+b = 0$ able to separate (classify) positive from negative examples, where $\vec{x}$ is the feature vector representation of an object o, and $\vec{w} \in \Re^n$ and $b \in \Re$ are parameters, learned from the training examples by applying the *Structural Risk Minimization* principle [27]. The object o is mapped in $\vec{x}$ with a feature function $\phi : \mathcal{O} \to \Re^n$, where $\mathcal{O}$ is the set of objects. o is categorized in the target class only if $H(\vec{x}) \geq 0$.

The kernel trick allows us to rewrite the decision hyperplane as:

$$H(\vec{x}) = \Big(\sum_{i=1..l} y_i\alpha_i\vec{x}_i\Big)\cdot\vec{x}+b =$$
$$\sum_{i=1..l} y_i\alpha_i\vec{x}_i\cdot\vec{x}+b = \sum_{i=1..l} y_i\alpha_i\phi(o_i)\cdot\phi(o)+b.$$

where, y_i is equal to 1 for positive and -1 for negative examples, $\alpha_i \in \Re$ with $\alpha_i \geq 0$, and o_i $\forall i \in \{1,..,l\}$ are the training instances. The product $K(o_i, o) = \langle\phi(o_i)\cdot\phi(o)\rangle$ is the kernel function associated with the mapping ϕ.

Note that we do not need to actually apply the mapping ϕ, since we can use $K(o_i, o)$ directly. This allows us, under the Mercer's conditions [25], to define abstract kernel functions which generate implicit feature spaces. An interesting example is given by the polynomial kernel: $PK(o_1, o_2) = (c + \vec{x}_1\cdot\vec{x}_2)^d$, where c is a constant and d is the degree of the polynomial. This kernel generates the space of all conjunctions of feature groups up to d elements.

In our relational data mining framework, we consider objects organized in a tree structure, thus the tree kernels described in the next section are used to classify relations and roles.

3.2 Tree Kernels

Tree kernels represent trees in terms of their substructures (fragments). When comparing two trees T_1 and T_2, the kernel function detects if a tree subpart common to both trees belongs to the feature space that we intend to generate. For such purpose, the desired fragments need to be described. We consider three important characterizations: the subtrees (STs), the subset trees (SSTs) and the partial trees (PTs).

A *subtree* (ST) is any node of a tree along with all its descendants. For example, Figure 4(a) shows the syntactic parse tree of the sentence *"Mary brought a cat"* along with its 6 STs.

A *subset tree* (SST) is a more general structure since its leaves can be non-terminal symbols. For example, Figure 4(b) shows 10 SSTs (out of 17) of the subtree in Figure 4(a) rooted in `VP`. The SSTs satisfy the constraint that grammatical rules cannot be broken. For example, `[VP [V NP]]` is an SST which has two non-terminal symbols, `V` and `NP`, as leaves whereas `[VP [V]]` is not an SST.

If we relax such constraint over the SSTs, we obtain more general substructures called *partial trees* (PTs). These can be generated by the application of partial production rules of the grammar. Consequently, `[VP [V]]` and `[VP [NP]]` are valid PTs. Figure 4(c) shows that the number of PTs derived from the same tree as before is still higher (i.e. 30 PTs).

The main idea of tree kernels is to compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. In the following paragraphs, the equation for the efficient evaluation of ST, SST and PT kernels are reported.

To evaluate the above kernels between two trees T_1 and T_2, we need to define a set $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$, i.e. a tree fragment space, and an indicator function $I_i(n)$, equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over T_1 and T_2 is $TK(T_1, T_2) = \sum_{n_1\in N_{T_1}}\sum_{n_2\in N_{T_2}}\Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1)I_i(n_2)$. The latter is equal to

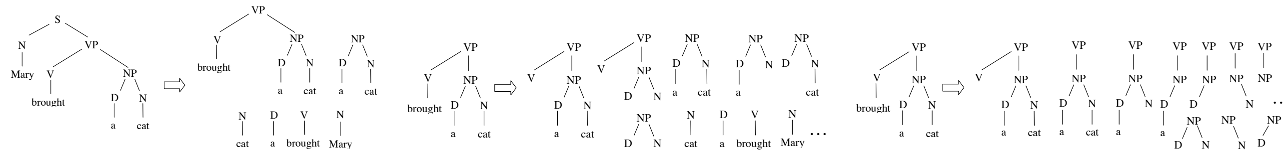


Figure 4. Examples of different classes of tree fragments.

the number of common fragments rooted in the n_1 and n_2 nodes.

The Δ function depends on the type of fragments that we consider as *basic* features. For example, to evaluate the fragments of type ST or SST, it can be defined as:

1. if the productions at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children (i.e. they are pre-terminals symbols) then $\Delta(n_1, n_2) = 1$;
3. if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (\sigma + \Delta(c_{n_1}^j, c_{n_2}^j)) \qquad (1)$$

where $\sigma \in \{0, 1\}$, $nc(n_1)$ is the number of children of n_1 and c_n^j is the j-th child of the node n. Note that, since the productions are the same, $nc(n_1) = nc(n_2)$.

When $\sigma = 0$, $\Delta(n_1, n_2)$ is equal 1 only if $\forall j \;\; \Delta(c_{n_1}^j, c_{n_2}^j) = 1$, i.e. all the productions associated with the children are identical. By recursively applying this property, it follows that the subtrees in n_1 and n_2 are identical. Thus, Eq. 1 evaluates the subtree (ST) kernel. When $\sigma = 1$, $\Delta(n_1, n_2)$ evaluates the number of SSTs common to n_1 and n_2 as proved in [5].

Moreover, a decay factor λ can be added by modifying steps (2) and (3) as follows[1]:

2. $\Delta(n_1, n_2) = \lambda$,
3. $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (\sigma + \Delta(c_{n_1}^j, c_{n_2}^j))$.

The computational complexity of Eq. 1 is $O(|N_{T_1}| \times |N_{T_2}|)$ but as shown in [17], the average running time is linear, i.e. $O(|N_{T_1}| + |N_{T_2}|)$.

PTFs have been defined in [17]. Their computation is carried out by the following Δ function:

1. if the node labels of n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. else $\Delta(n_1, n_2) =$
   $1 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1)=l(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))$

where $\vec{I}_1 = \langle h_1, h_2, h_3, ..\rangle$ and $\vec{I}_2 = \langle k_1, k_2, k_3, ..\rangle$ are index sequences associated with the ordered child sequences c_{n_1} of n_1 and c_{n_2} of n_2, respectively, $\vec{I}_{1j}$ and $\vec{I}_{2j}$ point to the j-th child in the corresponding sequence, and, again, $l(\cdot)$ returns the sequence length, i.e. the number of children.

Furthermore, we add two decay factors: μ for the depth of the tree and λ for the length of the child subsequences with respect to the original sequence, i.e. we account for gaps. It follows that $\Delta(n_1, n_2) =$

$$\mu\Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1)=l(\vec{I}_2)} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big), \quad (2)$$

where $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11}$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21}$. This way, we penalize both larger trees and child subsequences with gaps. Equation 2 is a more general one, the kernel can be applied to PTs. Also note that, if we only consider the contribution of the longest child sequence from node pairs that have the same children, we actually implement the SST kernel. For the ST computation, we also need to remove the λ^2 term from Eq. 2.

3.3 Combining and engineering kernel functions

Tree kernels can be combined with other kernels, for example the polynomial kernel over standard feature vectors, by summing or multiplying them. Another important aspect is the engineering property of tree kernels which allows to obtain efficient and accurate feature spaces by simply extracting subparts of the initial input tree. For example, if the structure of our data is a tree containing hundreds of thousands of nodes, the kernel computation would be very expensive in terms of time and memory occupancy.

In such conditions, we can assume that nodes located very far in the structures are independent and we may consider the subtree which only includes a target set of nodes. For example, in case of relation extraction from texts it is convenient to use the subtree in Figure 3.d instead of the whole tree of the frame (Figure 3.a) to classify the relation between the target and the argument, *"its title"*. Such

[1] To have a similarity score between 0 and 1, we also apply the normalization in the kernel space, i.e.: $K'(T_1, T_2) = \frac{TK(T_1,T_2)}{\sqrt{TK(T_1,T_1) \times TK(T_2,T_2)}}$.

subtree, called Argument Spanning Tree (AST_1) [20] is obtained by considering the minimum subtree that covers the target and only one argument node, along with their descendants. An AST_1 can be regarded as a subset of a larger structure, the AST_n, which is defined as the minimum tree that spans all the arguments that take part in a relation [20]. The subfigures labeled (b) and (c) in Figure 3 show the AST_n corresponding to the predicates encoded in the example sentence (a).

4. Previous work

Semantic Role Labeling is a broadly employed text mining technique, as it allows for the addition of structured semantic information to plain text [23]. The automatically extracted patterns can be eventually used to discover new relations as well as to access the encoded information more conveniently. [3] and more recently [16] present good overviews on state-of-the-art systems for SRL.

As a straightforward applied scenario in the domain of biology and medicine is the rich (and inherently textual) scientific literature that can be processed with automatic tools in order to discover new hints about protein interaction or gene functions. For example, in [2] an SRL system is used to automatically extract protein transport information. The system, based on word chunks, uses SVMs as its learning framework. It is generally accurate and it also shows good results on automatically identified proteins, unlike traditional rule-based approaches which are generally less robust towards new phenomena.

Our framework of relation extraction from structured data can be extended to other application domains. As an example, given the popularity of the format across many diverse communities, a great deal of attention is devoted to relation extraction from XML documents [29].

5. Experiments

In this section, we report extensive experimentation on mining semantic patterns from texts in the form of predicate argument structures. In the computational linguistics community, such task is often referred to as Semantic Role Labeling. When extended to multi-frame scenarios, it is referred to as Frame Recognition. Since our approach is inherently supervised, we concentrate on the PropBank [21] and FrameNet [1] corpora. In fact, they allow for the evaluation of our models against fairly large amounts of annotated data, spanning different linguistic domains and target semantic relations. For both corpora, our input data consist of automatically generated parse trees of natural language sentences, along with and human-annotated target predicates and roles. In more detail, the syntactic structure of the above linguistic objects is the parse tree automatically generated by means of the Charniak's constituency based parser [4].

We exploited the architecture shown in Figure 2 over the mentioned corpora, in which feature representations are provided by structural kernels. Additionally, we exploited the features manually designed by computational linguists for SRL systems in the last decade, including the *Path*, *Node Type*, *Head Word*, *First and Last Word Part Of Speech* features [10, 22]. In this way, we made available to our learning machines different combinations of polynomial kernels with the Tree Kernels described in Section 3.2.

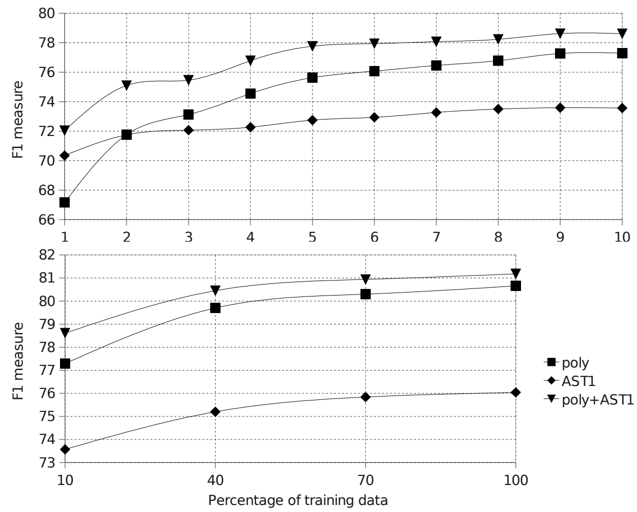


Figure 5. Learning curve for the argument selection task.

We used the SVM-Light implementation [13] of the SVM algorithm with the default regularization parameter (option $-c$) and $\lambda = 0.4$.

In the remainder, Section 5.1 details the setup and results of our experiments on the PropBank dataset, whereas Section 5.2 will focus on FrameNet.

5.1 Evaluation on PropBank corpus

Since the CoNLL'05 shared task [3], the Proposition Bank [21] has been a major benchmark for the evaluation of supervised models for SRL. It consists of 43,616 sentences and 99,242 predicate argument structures organized into 24 *sections* which are annotated on the top of handcrafted syntactic parse trees. As a common experiment setting established in the SRL community, sections 02-21 are available for training, section 24 (1,347 sentences containing 3,247 predicate argument annotations) is used for development, and section 23 (2,417 sentences containing 5,267 predicate argument annotations) for testing. The target objects are verbs, and the role set consists of 59 distinct labels, which are shared across different verbs, although most of them are defined on a per-predicate basis. In order to allow systems to be trained on automatic parse trees, the shared task organizers provided a mapping between the annotations defined on the handcrafted trees and the automatic parses generated by Charniak's parser on the corpus sentences.

To evaluate the accuracy of our relational miner on node selection (NS), we ran a set of experiments using 1,000,000 candidate arguments for training, from sections 2 to 6. We used 3 kernel combinations: *poly*, a polynomial kernel of degree 3 on a vector of manually designed features; the *SST Tree Kernel on AST_1 structures* (shortly SST, see Section 3.3); and SST+*poly*, an additive combination of the two previous kernels. Figure 5 shows the F_1-measure (i.e. the harmonic mean between Precision and Recall) achieved by different kernel configurations on the candidate arguments of section 24 (149,140 candidate examples after filtering) when varying the percentage of training data.

The plot shows that SST[2] improves on *poly* by about 3 percent points when very few training data (i.e. 10,000 instances) are used. When all the available data are used for learning, the polynomial kernel outperforms SST by about 5 points. This limited loss of performance is a very good result, considering that SST only encodes structural information without relying on the properties of the considered objects.

The relatively higher F_1 of the polynomial kernel is traded for the cost of manually designing features. In fact, those used in the experiments have been developed in several years of study by expert computational linguists. Clearly, such features are very useful and, when available, they should be combined with tree kernels to further increase the model accuracy. Indeed, the combined kernel (SST+*poly*) always outperforms the individual configurations, as it is able to conveniently represent both object-specific and structural information. Using one million training instances, the SST+*poly* kernel classifies the instances of section 24 with a Precision of 81.64%, a Recall of 80.73% and an F_1 measure of 81.18. This achieves the best result obtained in CoNNL 2005 when no classifier committee and no multiple syntactic parsers are used [18].

task	P	R	F_1
NS	82.23%	80.83%	81.52
NS+RC	76.55%	75.24%	75.89
Joint inference	80.16%	74.54%	77.25

Table 1. Results on the PropBank dataset.

Table 1 shows the results for the different SRL sub-tasks on the 269,888 candidate arguments of section 23. For NS and RC, the best model, SST+*poly*, was used. Moreover, we built a joint model based on the PT Kernel (Section 3.2) which combines the individual AST_1 structures into AST_n-like structures (as shown in Figure 3.b and 3.c, see also Section 3.3). These, by encoding the whole automatic annotation (role labels and predicate), are able to capture the global argument interdependencies [19]. This model achieves F_1=77.25 (last line in Table 1) which is near the state-of-the-art of SRL systems, e.g. [22].

[2] In the plot, SST is indicated as AST_1

5.2 Evaluation on FrameNet corpus

A natural application setting for the multi-frame mining architecture (or combined RMAs, Section 2.3) is the FrameNet lexical resource for English [9]. FrameNet is an ongoing lexicographic project based on Frame Semantics [1], which currently produced more than 135,000 sentences annotated on the basis of more than 800 frames, where each frame defines its local set of semantic roles.

For example, the sentence *"As a result of your win I can buy something special for your ma"* is annotated as an instance of the COMMERCE_SCENARIO frame, which includes frame elements (roles) as BUYER (*I*), GOODS (*something special*) and RECIPIENT (*for your ma*). Although both PropBank and FrameNet encode the relation between syntax and semantics [11], FrameNet sentences are not associated with human-validated syntactic trees. Therefore, semantic roles are annotated directly on the bare text. As a consequence, only automatic syntactic analysis is available for FrameNet. This constitutes an additional challenge for automatic frame and role detection, due to the high number of mismatches between the human-annotated semantic roles and the automatically-annotated syntactic constituents.

We applied our multi-frame RMA architecture to the supervised machine learning task of recognizing roles over free text sentences.

5.2.1 Multi-frame architecture configuration

The multi-frame architecture was configured for this task in the following way: first, we instantiated a specific frame model, i.e. a single RMA as in Figure 2, for each FrameNet frame. Recall that each RMA exploits two different machine learning models its two labeling stages, that is node selection (NS) and role classification (RC). We just considered those 502 frames actually populated with annotated sentences.

Second, we trained 5 different NS models for the 5 main categories of the target words (i.e. the syntactic categories of possible predicates[3]). This means that 5 binary classification models were learned over 782 frames (obtained when sentences in the above 502 frames are further partitioned by

[3] Verbal as well as nominal, adjectival, adverbial and prepositional predicates are defined in FrameNet.

	poly			SST			SST + poly			SST-L			SST-L + poly		
Eval setting	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
NS (nodes)	.887	.675	.767	.949	.652	.773	.915	.698	**.792**	.938	.659	.774	.908	.701	.791
NS (words)	.850	.647	.735	.919	.631	.748	.875	.668	**.758**	.906	.636	.747	.868	.670	.757
NS+RC (nodes)	.654	.498	.565	.697	.479	.568	.680	.519	**.588**	.689	.484	.569	.675	.521	.588
NS+RC (words)	.625	.476	.540	.672	.462	.548	.648	.495	**.561**	.663	.466	.547	.644	.497	.561

Table 2. Results on FrameNet: kernel classifiers with 2% training data for NS and 90% for RC.

part of speech of their predicates), where each model predicts the role/non-role class.

Third, for each syntactic word category and for each frame, we learned a multi-role classifier, obtaining 782 different one-versus-all multi-classification models, collectively composed by 5,345 binary classifiers (one for each role).

Finally, we applied this multi-frame setting for recognizing argument nodes as well as their semantic roles in the FrameNet sentences, where the frame label and the target predicate were considered as given.

5.2.2 Experiment setting and results

The Version 1.3 of FrameNet[4] was used for both learning and test. After preprocessing and parsing with the Charniak's parser[5], we obtained 135,293 annotated and parsed sentences. We split the data considering the part of speech of predicates, ending up with 782 different frames.

The overall dataset was partitioned into three subsets. We used a 2% of data (2,782 sentences) as NS training set, 90% (121,798 sentences) as RC training set, and 1% (1,345 sentences) as overall test set. All of these subsets are disjoint.

We also report the number of positive and negative training examples provided to our binary SVM-based classifiers. For NS, we used: 2,764 positive and 37,497 negative examples for verbal predicates, 1,189 and 35,576 for nominal, 615 and 14,544 for adjectival, 0 and 40 for adverbial, and 7 and 177 for prepositional predicates. The total examples for NS were 4,575 and 87,834. For RC, the total numbers were 207,662 and 1,960,423, which divided by the number of role labels shows the average number of 39 positive versus 367 negative examples per role.

We tested several kernels over standard [10, 22] and structured (AST_1) features [20]: the polynomial kernel (*poly*, with a degree of 3), the subset tree kernel (SST), and the SST kernel combined with the bag-of-word kernel on the tree leaves (SST-L). Also, the combinations of SST and SST-L with *poly* were tested.

Table 2 reports Precision, Recall and F_1 measure of the above classifiers over different tasks. The 4 rows in the table show in turn: (1) the "pure" performance of the BD classifier, i.e. considering correct the classification decisions also when a correctly classified tree *node* does not exactly correspond to a valid sentence constituent. Such mismatch frequently happens when the parse tree (which is automatically generated) includes incorrect nodes and attachments (also see the initial discussion in Section 5.2); (2) the performance of the BD classification "projected" on the tree leaves, i.e. when matching not only the constituent node as in 1, but also the selected *words* (leaves) with those in the FrameNet gold standard. This implies an exact syntactic analysis being encoded in the subtree; (3) the same as 1, with the argument role classification (RC) also performed (i.e. Frame Element labels must match as well); (4) the same as 2, with RC also performed.

The results improve when the amount of training data for the NS model is also increased from 2% to 90%. As shown in Table 3, the SST+*poly* kernel achieves 1.0 Precision, 0.732 Recall and 0.847 F_1 on NS. These figures can be compared to 0.855 Precision, 0.669 Recall and 0.751 F_1 of the system described in [8], achieved with the same amount of training data. In conclusion, our best learning scheme is currently capable of tagging FrameNet data from noisy syntax with exact boundaries and role labels at 63% F_1. Our next steps will be first, further improving the RC models exploiting FrameNet-specific information (such as frame and role inheritance), and second, introducing an effective frame classifier to automatically choose Frame labels.

Enhanced SST + poly			
Eval Setting	P	R	F_1
NS (nodes)	1.0	.732	.847
NS (words)	.963	.702	.813
NS+RC (nodes)	.784	.571	.661
NS+RC (words)	.747	.545	.630

Table 3. Results on FrameNet. SST+poly with 90% training data for NS and RC.

6. Conclusions

The extraction of relational patterns from structured data is a relevant topic within the DM community. A general framework able to cope with this kind of data may handle

[4]Only its lexicographic data were used, leaving out the continuous annotation texts.

[5]A few sentences were discarded in this step due to parsing problems.

the growing amount of information which is naturally available in a structured way. Furthermore, by using automatic text processing tools such as constituency or dependency parsers, it would be possible to convert textual information into structured one, and to use the same framework to mine patterns in originally unstructured documents.

We presented a framework for relational mining over structured data capable of scaling to different tasks and domains. This flexibility is achieved by means of Tree Kernels and structured features, which allow for the encoding of structural information directly into the learning algorithm. To assess the accuracy of our approach, we executed a set of experiments on Semantic Role Labeling and Frame Recognition, over the two established PropBank and FrameNet corpora.

On both tasks, our system achieves state of the art accuracy. Also, Frame Recognition achieves good accuracy with very small training sets. Especially in such difficult conditions, the impact of Tree Kernels and structured features is noticeable and relevant for real-world tasks.

Acknowledgments

This research is partially supported by the LiveMemories Project funded by the *Provincia Autonoma di Trento* (PAT). The authors wish to thank the anonymous reviewers for their helpful comments.

References

[1] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet project. In *Proceedings of COLING-ACL '98*, pages 86–90, 1998.

[2] S. Bethard, Z. Lu, J. H. Martin, and L. Hunter. Semantic role labeling for protein transport predicates. *BMC Bioinformatics*, 9:277+, June 2008.

[3] X. Carreras and L. Màrquez. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*, pages 152–164, Ann Arbor, Michigan, June 2005.

[4] E. Charniak. A maximum-entropy-inspired parser. In *Proceedings of NAACL 2000*, San Francisco, CA, USA, 2000.

[5] M. Collins and N. Duffy. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete structures, and the voted perceptron. In *ACL02*, pages 263–270, 2002.

[6] S. Dzeroski and H. Blockeel. Introduction to the workshop. In *MRDM '05: Proceedings of the 4th international workshop on Multi-relational mining*, New York, NY, USA, 2005. ACM.

[7] S. Dzeroski and N. Lavrac, editors. *Relational Data Mining*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[8] K. Erk and S. Pado. Shalmaneser - a flexible toolbox for semantic role assignment. In *Proceedings of LREC 2006*, Genoa, Italy, 2006.

[9] C. J. Fillmore. The Case for Case. In E. Bach and R. T. Harms, editors, *Universals in Linguistic Theory*, pages 1–210. Holt, Rinehart, and Winston, New York, 1968.

[10] D. Gildea and D. Jurafsky. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288, 2002.

[11] A.-M. Giuglea and A. Moschitti. Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of ACL 2006*, Sydney, Australia, 2006.

[12] A. Haghighi, K. Toutanova, and C. Manning. A joint model for semantic role labeling. In *Proceedings of CoNLL-2005*, Ann Arbor, Michigan, June 2005.

[13] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184, 1999.

[14] J. D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. Genia corpus–semantically annotated corpus for bio-textmining. *Bioinformatics*, 19 Suppl 1, 2003.

[15] O. Kolak and B. N. Schilit. Generating links by mining quotations. In *Proceedings of the 9th ACM conference on Hypertext and hypermedia*, 2008.

[16] L. Marquez, X. Carreras, K. C. Litkowski, and S. Stevenson. Semantic role labeling: An introduction to the special issue. *Computational Linguistics*, 34(2):145–159, 2008.

[17] A. Moschitti. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *Proceedings of ECML 2006*, pages 318–329, Berlin, Germany, 2006.

[18] A. Moschitti, B. Coppola, A. Giuglea, and R. Basili. Hierarchical semantic role labeling. In *Proceedings of the CoNLL 2005 shared task on SRL*, Ann Arbor, Michigan, 2005.

[19] A. Moschitti, D. Pighin, and R. Basili. Semantic role labeling via tree kernel joint inference. In *Proceedings of CoNLL-X*, New York City, 2006.

[20] A. Moschitti, D. Pighin, and R. Basili. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224, 2008.

[21] M. Palmer, D. Gildea, and P. Kingsbury. The Proposition Bank: an Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, 2005.

[22] S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. Support Vector Learning for Semantic Argument Classification. *Machine Learning*, 60:1-3:11–39, 2005.

[23] S. Pradhan, K. Hacioglu, W. Ward, J. Martin, and D. Jurafsky. Semantic role parsing: Adding semantic structure to unstructured text, 2003.

[24] R. Sanderson and P. Watry. Integrating data and text mining processes for digital library applications. In *Proceedings of JCDL 2007*, 2007.

[25] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[26] E. Tjong Kim Sang, S. Canisius, A. van den Bosch, and T. Bogers. Applying spelling error correction techniques for improving semantic role labelling. In *Proceedings of CoNLL-2005*, Ann Arbor, Michigan, June 2005.

[27] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[28] X. Zhou, H. Han, I. Chankai, A. Prestrud, and A. Brooks. Approaches to text mining for clinical medical records. In *Proceedings of SAC 2006*, 2006.

[29] X. Zhou, X. Pan, and Y. Ren. Web mining of relations from xml and construct database schema. In *CIMCA '06: Proceedings of CIMCA 2006*, 2006.