# Learning to Tag Text from Rules and Examples

Michelangelo Diligenti, Marco Gori, and Marco Maggini

Dipartimento di Ingegneria dell'Informazione, Università di Siena, Italy
{diligmic,marco,maggini}@dii.unisi.it

**Abstract.** Tagging has become a popular way to improve the access to resources, especially in social networks and folksonomies. Most of the resource sharing tools allow a manual labeling of the available items by the community members. However, the manual approach can fail to provide a consistent tagging especially when the dimension of the vocabulary of the tags increases and, consequently, the users do not comply to a shared semantic knowledge. Hence, automatic tagging can provide an effective way to complete the manual added tags, especially for dynamic or very large collections of documents like the Web. However, when an automatic text tagger is trained over the tags inserted by the users, it may inherit the inconsistencies of the training data. In this paper, we propose a novel approach where a set of text categorizers, each associated to a tag in the vocabulary, are trained both from examples and a higher level abstract representation consisting of FOL clauses that describe semantic rules constraining the use of the corresponding tags. The FOL clauses are compiled into a set of equivalent continuous constraints, and the integration between logic and learning is implemented in a multi-task learning scheme. In particular, we exploit the kernel machine mathematical apparatus casting the problem as primal optimization of a function composed of the loss on the supervised examples, the regularization term, and a penalty term deriving from forcing the constraints resulting from the conversion of the logic knowledge. The experimental results show that the proposed approach provides a significant accuracy improvement on the tagging of bibtex entries.

## 1 Introduction

Tagging consists in associating a set of terms to resources (e.g. documents, pictures, products, blog posts, etc.) with the aim of making it easier their search and organization. Tags reflecting semantic properties of the resources (e.g. categories, keywords summarizing the content, etc.) are effective tools for searching the collection. Therefore, high consistency and precision in the tag assignment would allow the development of more sophisticated information retrieval mechanisms, as the ones typically provided in search-by-keyword applications. In the context of folksonomies or Web directories, tagging is manually performed and the vocabulary of tags is usually unrestricted and freely chosen by the users. Beside semantic tags, the users often use tags to represent meta–information to be attached to each item (e.g. dates, the camera brand for pictures, etc.). Unfortunately, a manual collective tagging process has many limitations. First, it

is not suited for very large and dynamic collections of resources, where response time is crucial. Furthermore, the collective tagging process does not provide any guarantee of consistency of the tags across the different items, creating many issues for the subsequent use of the tags [8]. This problem is especially crucial in the context of social networks and folksonomies where there is not a standardized semantic knowledge shared by the users, since tags are independently chosen by each user without any restriction.

Automatic text tagging is regarded as a way to address, at least partially, these limitations [7]. Automatic text tagging can be typically seen as a classical text categorization task [10], where each tag corresponds with a different category. Differently to many categorization tasks explored in the literature, the number of tags is typically in the order of hundreds to several thousands, and the tags are not mutually exclusive, thus yielding a multi-class classification task. Automatic text categorization and tagging has been approached with either pure ontology-based approaches [12, 6] or with learning-from-examples techniques based on machine learning [1, 11].

Manually inserted tags can be used to effectively train an automatic tagger, which generalizes the tags to new documents [9]. However, when an automatic text tagger is trained over the tags inserted by the users of a social network, it may inherit the same inconsistencies of the training data.

The approach presented in this paper bridges the knowledge-based and machine learning approaches, where a text categorizer and the reasoning process defined via a formal logic language are jointly implemented via kernel machines. The formal logic language enforces the tag consistency without depending on specially trained human taggers. In particular, higher level abstract representations consist of FOL clauses that constrain the configurations assumed by the task predicates, each one associated to a tag. The FOL clauses are then compiled into a set of equivalent continuous constraints, and the integration between logic and learning is implemented in a multi-task learning scheme where the kernel machine mathematical apparatus makes it possible to cast the problem as primal optimization of an objective function combining the loss on the supervised examples, the regularization term, and a penalty term deriving from forcing the constraints resulting from the conversion of the FOL knowledge base.

One main contribution of this paper is the definition of a novel approach to convert the FOL clauses into a set of constraints. Unlikely previously solutions, the proposed conversion process guarantees that each configuration satisfying the FOL rules corresponds to a minimum of the cost function. Furthermore, the paper provides an experimental evaluation of the proposed technique on a real world text tagging dataset. The paper is organized as follows. The next section describes how constraints can be embedded as penalties in the kernel machine framework. Section 3 shows how constraints provided as a FOL knowledge base can be mapped to a set of continuous penalty functions. Finally, section 4 reports the experimental results on a bibtex tagging benchmark.

## 2 Learning to tag with constraints

We consider an alphabet $T$ of tags, whose size we indicate as $|T|$. Therefore, a set of multivariate functions $\{f_k(\boldsymbol{x}), k = 1, \ldots, |T|\}$ must be estimated, where the $k$-th function approximates a predicate determining whether the $k$-th tag should be assigned to the input text represented by the feature vector $\boldsymbol{x} \in \mathcal{D}$. We consider the case when the task functions $f_k$ have to meet a set of functional constraints that can be expressed as

$$\phi_h(f_1, \ldots, f_{|T|}) = 0 \quad h = 1, \ldots, H \tag{1}$$

for any valid choice of the functions $f_k : k = 1, \ldots, |T|$ defined on the input domain $\mathcal{D}$. In particular, in the next section we will show how appropriate functionals can be defined to force the function values to meet a set of first-order logic constraints.

Once we assume that all the functions $f_k$ can be approximated from an appropriate Reproducing Kernel Hilbert Space $\mathcal{H}$, the learning procedure can be cast as a set of $|T|$ optimization problems that aim at computing the optimal functions $f_1, \ldots, f_{|T|}$ in $\mathcal{H}$. In the following, we will indicate by $\boldsymbol{f} = [f_1, \ldots, f_{|T|}]'$ the vector collecting these functions. In general, it is possible to consider a different RKHS for each function given some a priori knowledge on its properties (i.e. we may decide to exploit different kernels for the expansion).

We consider the classical learning formulation as a risk minimization problem. Assuming that the correlation among the input features $\boldsymbol{x}$ and the desired $k$-th task function output $y$ is provided by a collection of supervised input-target examples

$$\mathcal{L}_k = \left\{ \left( \boldsymbol{x}_k^i, y_k^i \right) | i = 1, \ldots, \ell_k \right\} , \tag{2}$$

we can measure the risk associated to a hypothesis $\boldsymbol{f}$ by the empirical risk

$$R[\boldsymbol{f}] = \sum_{k=1}^{|T|} \frac{\lambda_k^e}{|\mathcal{L}_k|} \sum_{\left( \boldsymbol{x}_k^j, y_k^j \right) \in \mathcal{L}_k} L^e \left( f_k(\boldsymbol{x}_k^j), y_k^j \right) \tag{3}$$

where $\lambda_k^e > 0$ is the weight of the risk for the $k$-th task and $L^e \left( f_k(\boldsymbol{x}), y \right)$ is a loss function that measures the fitting quality of $f_k(\boldsymbol{x})$ with respect to the target $y$. Common choices for the loss function are the quadratic function especially for regression tasks, and the hinge function for binary classification tasks. Different loss functions and $\lambda_k^e$ parameters may be employed for the different tasks.

As for the regularization term, unlike the general setting of multi-task kernels [2], we simply consider scalar kernels that do not yield interactions amongst the different tasks, that is

$$N[\boldsymbol{f}] = \sum_{k=1}^{|T|} \lambda_k^r \cdot ||f_k||_{\mathcal{H}}^2, \tag{4}$$

where $\lambda_k^r > 0$ can be used to weigh the regularization contribution for the $k$-th task function.

Clearly, if the tasks are uncorrelated, the optimization of the objective function $R[\boldsymbol{f}] + N[\boldsymbol{f}]$ with respect to the $|T|$ functions $f_k \in \mathcal{H}$ is equivalent to $|T|$ stand-alone optimization problems. However, if we consider the case in which some correlations among the tasks are known a priori and coded as rules, we can enforce also these constraints in the learning procedure. Following the classical penalty approach for constrained optimization, we can embed the constraints by adding a term that penalizes their violation. In general we can assume that the functionals $\phi_h(\boldsymbol{f})$ are strictly positive when the related constraint is violated and zero otherwise, such that the overall degree of constraint violation of the current hypothesis $\boldsymbol{f}$ can be measured as

$$V[\boldsymbol{f}] = \sum_{h=1}^{H} \lambda_h^v \cdot \phi_h(\boldsymbol{f}) \ , \tag{5}$$

where the parameters $\lambda_h^v > 0$ allow us to weigh the contribution of each constraint. It should be noticed that, differently from the previous terms considered in the optimization objective, the penalty term involves all the functions and, thus, explicitly introduces a correlation among the tasks in the learning statement. In general, the computation of the functionals $\phi_h(\boldsymbol{f})$ implies the evaluation of some property[1] of the functions $f_k$ on their whole input space $\mathcal{D}$. However, we can approximate the exact computation by considering an empirical realization of the input distribution. We assume that beside the labeled examples in the supervised sets $\mathcal{L}_k$, a collection of unsupervised examples $\mathcal{U} = \{\boldsymbol{x}^i | i = 1, \ldots, u\}$ is also available. If we define the set $\mathcal{S}_k^L = \{\boldsymbol{x}_k | (\boldsymbol{x}_k, y_k) \in \mathcal{L}_k\}$ containing the inputs for the labeled examples for the $k$-th task, in general we can assume that $\mathcal{S}_k^L \subset \mathcal{U}$, i.e. all the available points are added to the unsupervised set. Even if this is not always required, it is clearly reasonable when the supervisions are partial, i.e. a labeled example for a task $k$-th is not necessarily contained in the supervised learning sets for the other tasks. This assumption is important for tagging tasks, where the number of classes is very high and providing an exhaustive supervision over all the classes is generally impossible. Finally, the exact constraint functionals will be replaced by their approximations $\hat{\phi}_h(\mathcal{U}, \boldsymbol{f})$ that exploit only the values of the unknown functions $\boldsymbol{f}$ computed for the points in $\mathcal{U}$. Therefore, in eq. (5) we can consider $\phi_h(\boldsymbol{f}) \approx \hat{\phi}_h(\mathcal{U}, \boldsymbol{f})$, such that the objective function for the learning algorithm becomes

$$E[\mathbf{f}] = R[\mathbf{f}] + N[\mathbf{f}] + \sum_{h=1}^{H} \lambda_h^v \hat{\phi}_h(\mathcal{U}, \boldsymbol{f}) \ . \tag{6}$$

The solution to the optimization task defined by the objective function of equation (6) can be computed by considering the following extension of the Representer Theorem.

[1] As shown in the following, the functional may imply the computation of the function maxima, minima, or integral on the input domain $\mathcal{D}$.

**Theorem 1.** *Given a multitask learning problem for which the task functions* $f_1, \ldots, f_{|T|}$, $f_k : \mathbb{R}^d \rightarrow \mathbb{R}$, $k = 1, \ldots, |T|$, *are assumed to belong to the Reproducing Kernel Hilbert Space* $\mathcal{H}$ *and the problem is formulated as*

$$[f_1^*, \ldots, f_{|T|}^*] = argmin_{f_k \in \mathcal{H}, k=1,\ldots,|T|} E[f_1, \ldots, f_{|T|}]$$

*where* $E[f_1, \ldots, f_{|T|}]$ *is defined as in equation (6), then each function* $f_k^*$ *in the solution can be expressed as*

$$f_k^*(\boldsymbol{x}) = \sum_{\boldsymbol{x}^i \in \mathcal{S}_k} w_{k,i}^* K(\boldsymbol{x}^i, \boldsymbol{x})$$

*where* $K(\boldsymbol{x}', \boldsymbol{x})$ *is the reproducing kernel associated to the space* $\mathcal{H}$, *and* $\mathcal{S}_k = \mathcal{S}_k^L \cup \mathcal{U}$ *is the set of the available samples for the k-th task function.*

The proof is a straightforward extension of that for the original Representer Theorem and is based on the fact that the objective function only involves values of the functions $f_k$ computed on the finite set of supervised and unsupervised points.

## 3 Translation of first-order logic clauses into real-valued constraints

We consider knowledge-based descriptions given by first-order logic (FOL–KB). While the framework can be easily extended to arbitrary FOL predicates, in this paper we will focus on formulas containing only unary predicates to keep the notation simple. In the following, we indicate by $\mathcal{V} = \{v_1, \ldots, v_N\}$ the set of the variables used in the KB, with $v_s \in \mathcal{D}$. Given the set of predicates used in the KB $\mathcal{P} = \{p_k | p_k : \mathcal{D} \rightarrow \{true, false\},\ k = 1, \ldots, |T|\}$, the clauses will be built from the set of atoms $p(v) : p \in \mathcal{P}, v \in \mathcal{V}$.

Any FOL clause has an equivalent version in *Prenex Normal form* (PNF), that has all the quantifiers $(\forall, \exists)$ and their associated quantified variables at the beginning of the clause. Standard methods exist to convert a generic FOL clause into its corresponding PNF and the conversion can be easily automated. Therefore, with no loss of generality, we restrict our attention to FOL clauses in the PNF form. We assume that the task functions $f_k$ are exploited to implement the predicates in $\mathcal{P}$ and that the variables in $\mathcal{V}$ correspond to the input $\boldsymbol{x}$ on which the functions $f_k$ are defined. Different variables are assumed to refer to independent values of the input features $\boldsymbol{x}$. In this framework, the predicates yield a continuous real value that can be interpreted as a *truth degree*. The FOL–KB will contain a set of clauses corresponding to expressions with no free variables (i.e. all the variables appearing in the expression are quantified) that are assumed to be *true* in the considered domain. These clauses can be converted into a set of constraints as in eq. (1) that can be enforced during the kernel based learning process. The conversion process of a clause into a constraint functional consists of the following steps:

I. Conversion of the Propositional Expression: conversion of the quantifier-free expression using a mixture of Gaussians.
II. Quantifier conversion: conversion of the universal and existential quantifiers.

**Logic expressions in a continuous form**

This subsection describes a technique for the conversion of an arbitrary propositional logic clause into a function, yielding a continuous truth value in $[0, 1]$. Previous work in the literature [3] concentrated on a conversion schema based on t-norms [5]. In this paper a different approach based on mixtures of Gaussians was pursued. This approach has the advantage of not making any independence assumption among the variables like it happens when using t-norms. In particular, let us assume to have available a propositional logic clause composed by $n$ logic variables. The logic clause is equivalent to its truth table containing $2^n$ rows, each one corresponding to a configuration of the input variables.

The continuous function approximating the clause is based on a set of Gaussian functions, each one centered on a configuration corresponding to a true value in the truth table. These Gaussians, basically corresponding to minterms in a disjunctive expansion of the clause, can be combined by summing all the terms as

$$t(x_1, \ldots, x_n) = \sum_{[c_1,\ldots,c_n]\in\mathcal{T}} \exp\left(-\frac{|[x_1, \ldots, x_n]' - [c_1, \ldots, c_n]'|^2}{2\sigma^2}\right) , \qquad (7)$$

where $x_1, \ldots, x_n$ are the input variables generalizing the Boolean values on a continuous domain, and $c_1, \ldots, c_n$ span the set $\mathcal{T}$ of all the possible configurations of the input variables which correspond to the true values in the table. In the following, we indicate as *mixture-by-sum* this conversion procedure.

Another possibility for combining the minterm functions is to select the one with the highest value for a given input configuration (*mixture-by-max*). This latter solution selects the configuration closest to the true value in the truth table as

$$t(x_1, \ldots, x_n) = \max_{[c_1,\ldots,c_n]\in\mathcal{T}} \exp\left(-\frac{|[x_1, \ldots, x_n]' - [c_1, \ldots, c_n]'|^2}{2\sigma^2}\right) . \qquad (8)$$

For example, let us consider the simple OR of two atoms $A \vee B$. The mixture-by-max corresponding to the logic clause is the continuous function $t : I\!R^2 \Rightarrow I\!R$

$$t(x_1, x_2) = \max(e^{-\frac{|[x_1,x_2]-[1,1]'|^2}{2\sigma^2}}, e^{-\frac{|[x_1,x_2]-[1,0]'|^2}{2\sigma^2}}, e^{-\frac{|[x_1,x_2]-[0,1]'|^2}{2\sigma^2}}) ,$$

where $\boldsymbol{x} = [x_1, x_2]$ collects the continuous values computed for the atoms A and B, respectively. Instead, for the mixture-by-sum, the clause is converted as

$$t(x_1, x_2) = e^{-\frac{|[x_1,x_2]-[1,1]'|^2}{2\sigma^2}} + e^{-\frac{|[x_1,x_2]-[1,0]'|^2}{2\sigma^2}} + e^{-\frac{|[x_1,x_2]-[0,1]'|^2}{2\sigma^2}} .$$

If $x_1, \ldots, x_n$ is equal to a configuration verifying the clause, in the case of the mixture-by-sum it holds $t(x_1, \ldots, x_n) \geq 1$ whereas for the mixture-by-max $t(x_1, \ldots, x_n) = 1$. Otherwise, the value of $t()$ will decrease depending on the distance from the closest configuration verifying the clause. The variance $\sigma^2$ is a parameter which can be used to determine how quickly $t(x_1, \ldots, x_n)$ decreases when moving away from a configuration verifying the constraint. Please note that each configuration verifying the constraint is always a global maximum of $t$ when using the mixture-by-max.

**Quantifier conversion**

The quantified portion of the expression is processed recursively by moving backward from the inner quantifier in the PNF expansion.

Let us consider the universal quantifier first. The universal quantifier requires that the expression must hold for any realization of the quantified variable $v_q$. When considering the real–valued mapping of the original boolean expression, the universal quantifier can be naturally converted measuring the degree of non-satisfaction of the expression over the domain $\mathcal{D}$ where the feature vector $\boldsymbol{x}$, corresponding to the variable $v_q$, ranges. In particular, it can be proven that if $E(v, \mathcal{P})$ is an expression with no quantifiers, depending on the variable $v$, and $t_E(v, \boldsymbol{f})$ is its translation into the mixture-by-max representation, given that $f_k \in \mathbb{C}^0,\ k = 1, \ldots, T$, then $\|1 - t_E(v, \boldsymbol{f})\|_p = 0$ if and only if $\forall v\ E(v, \mathcal{P})$ is *true*. Hence, in general, the satisfaction measure can be implemented by computing the overall distance of the penalty associated to the expression $E$, depending on the set of variables $\boldsymbol{v}_E$, i.e. $\varphi_E(\boldsymbol{v}_E, \boldsymbol{f}) = 1 - t_E(\boldsymbol{v}_E, \boldsymbol{f})$ for mixture-by-max, from the constant function equal to 0 (the value for which the constraint is verified), over the values in the domain $\mathcal{D}$ for the input $\boldsymbol{x}$ corresponding to the quantified variable $v_q \in \boldsymbol{v}_E$. In the case of the infinity norm we have

$$\forall v_q E(\boldsymbol{v}_E, \mathcal{P}) \rightarrow \|\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})\|_\infty = \sup_{v_q \in \mathcal{D}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})| \ , \tag{9}$$

where the resulting expression depends on all the variables in $\boldsymbol{v}_E$ except $v_q$. Hence, the result of the conversion applied to the expression $E_q(\boldsymbol{v}_{E_q}, \mathcal{P}) = \forall v_q\ E(\boldsymbol{v}_E, \mathcal{P})$ is a functional $\varphi_{E_q}(\boldsymbol{v}_{E_q}, \boldsymbol{f})$, assuming values in $[0, 1]$ and depending on the set of variables $\boldsymbol{v}_{E_q} = \boldsymbol{v}_E \setminus \{v_q\}$. The variables in $\boldsymbol{v}_{E_q}$ need to be quantified or assigned a specific value in order to obtain a constraint functional depending only on the functions $\boldsymbol{f}$.

In the conversion of the PNF representing a FOL constraint without free variables, the variables are recursively quantified until the set of the free variables is empty. In the case of the universal quantifier we apply again the mapping described previously. The existential quantifier can be realized by enforcing the De Morgan law to hold also in the continuous mapped domain. The De Morgan law states that $\exists v_q\ E(\boldsymbol{v}_E, \mathcal{P}) \Longleftrightarrow \neg \forall v_q\ \neg E(\boldsymbol{v}_E, \mathcal{P})$. Using the conversion of the universal quantifier defined in eq. (9), we obtain the following conversion for the existential quantifier

$$\exists v_q\ E(\boldsymbol{v}_E, \mathcal{P}) \rightarrow \inf\nolimits_{v_q \in \mathcal{D}} \varphi_E(\boldsymbol{v}_E, \boldsymbol{f}) \ .$$

The conversion of the quantifiers requires to extend the computation on the whole domain of the quantified variables. Here, we assume that the computation can be approximated by exploiting the available empirical realizations of the feature vectors. If we consider the examples available for training, both supervised and unsupervised, we can extract the empirical distribution $\mathcal{S}_k$ for the input to the $k$-th function. Hence, the universal quantifier exploiting the infinity norm is approximated as

$$\forall v_q \; E(\boldsymbol{v}_E, \mathcal{P}) \rightarrow \max_{v_q \in \mathcal{S}_{k(q)}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})| \; .$$

Similarly, for the existential quantifier it holds

$$\exists v_q \; E(\boldsymbol{v}_E, \mathcal{P}) \rightarrow \min_{v_q \in \mathcal{S}_{k(q)}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})| \; .$$

It is also possible to select a different functional norm to convert the universal quantifier. For example, when using the $\|\cdot\|_1$ norm, and the empirical distribution for the $\boldsymbol{x}$, we have

$$\forall v_q \; E(\boldsymbol{v}_E, \mathcal{P}) \rightarrow \frac{1}{|\mathcal{S}_{k(q)}|} \sum_{v_q \in \mathcal{S}_{k(q)}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})| \; .$$

Please note that $\varphi_E([], \boldsymbol{f})$ will always reduce to the following form, when computed for an empirical distribution of data for any selected functional norm

$$\varphi_E([], \boldsymbol{f}) = \mathrm{O}_{v_{s(1)} \in \mathcal{S}_{k(s(1))}} \ldots \; \mathrm{O}_{v_{s(Q)} \in \mathcal{S}_{k(s(Q))}} t_{E_0}(\boldsymbol{v}_{E_0}, \boldsymbol{f}) \; ,$$

where $E_0$ is the quantifier free expression, $\mathrm{O}_{v_q \in \mathcal{S}_{x_{j(q)}}}$ specifies the aggregation operator to be computed on the sample set $\mathcal{S}_{k(q)}$ for each quantified variable $v_q$. In the case of the infinity norm, $\mathrm{O}_{v_q \in \mathcal{S}_{k(q)}}$ is either the minimum or maximum operator over the set $\mathcal{S}_{k(q)}$. Therefore, the presented conversion procedure implements the logical constraint depending on the realizations of the functions over the data point samples. For this class of constraints, Theorem 1 holds and the optimal solution can be expressed as a kernel expansion over the data points. In fact, since the constraint is represented by $\varphi_E([], \boldsymbol{f}) = 0$ in the definition of the learning objective function of eq. (6) we can substitute $\hat{\phi}(\mathcal{U}, \boldsymbol{f}) = \varphi_E([], \boldsymbol{f})$.

When using the minimum and/or maximum operators for defining $\hat{\phi}(\mathcal{U}, \boldsymbol{f})$, the resulting objective function is continuous with respect to the parameters $w_{k,i}$ defining the RKHS expansion, since it is obtained by combining continuous functions. However, in general, its derivatives are no more continuous, but, in practice, this is not a problem for gradient descent based optimization algorithms once appropriate stopping criteria are applied. In particular, the optimal minima can be located also in configurations corresponding to discontinuities in the gradient values, i.e. when a maximum or minimum operator switches its choice among two different points in the dataset.

As an example of the conversion procedure, let $a(\cdot), b(\cdot)$ be two predicates, implemented by the functions $f_a(\cdot), f_b(\cdot)$. The clause $\forall v_1 \; \forall v_2 \; (a(v_1) \wedge \neg b(v_2)) \vee$

| |
|---|
| $\forall x$ phase(x) $\wedge$ transition(x) $\Rightarrow$ physics(x) |
| $\forall x$ chemistry(x) $\Rightarrow$ science(x) |
| $\forall x$ immunoelectrode(x) $\Rightarrow$ physics(x) $\vee$ biology(x) |
| $\forall x$ semantic(x) $\wedge$ web20(x) $\Rightarrow$ knowledgemanagement(x) |
| $\forall x$ rdf(x) $\Rightarrow$ semanticweb(x) |
| $\forall x$ software(x) $\wedge$ visualization(x) $\Rightarrow$ engineering(x) |
| $\forall x$ folksonomy(x) $\Rightarrow$ social(x) |
| $\forall x$ mining(x) $\wedge$ web(x) $\Rightarrow$ informationretrieval(x) |
| $\forall x$ mining(x) $\wedge$ information(x) $\Rightarrow$ datamining(x) |
| $\forall x$ computer(x) $\wedge$ science(x) $\Rightarrow$ engineering(x) |

Table 1: A sample of the semantic rules used in training the kernel machines.

$(\neg a(v_1) \wedge b(v_2))$ is converted starting with the conversion of the quantifier free expression $E_0([v_1, v_2], \{a(\cdot), b(\cdot)\}) = (a(v_1) \wedge \neg b(v_2)) \vee (\neg a(v_1) \wedge b(v_2))$ as

$$t_{E_0}([v_1, v_2], [f_a, f_b]) = \\ = \max\left(e^{-\frac{(f_a(v_1)-1)^2+f_b(v_2)^2}{2\sigma^2}}, e^{-\frac{f_a(v_1)^2+(f_b(v_2)-1)^2}{2\sigma^2}}\right) .$$

Then, the quantifier free expression $E_0([v_1, v_2], \{a(\cdot), b(\cdot)\})$ is converted into the distance measure and the two universal quantifiers are converted using the infinity norm over their empirical realizations, yielding the constraint

$$\varphi_{E_0}([], [f_a, f_b]) = \max_{v_1 \in \mathcal{S}_a} \max_{v_2 \in \mathcal{S}_b} \left(1 - t_{E_0}([v_1, v_2], [f_a, f_b])\right) = 0 .$$

## 4 Experimental results

The dataset used in the experiments consists of 7395 bibtex entries that have been tagged by users of a social network[2] using 159 tags. Each bibtex entry contains a small set of textual elements representing the author, the title, and the conference or journal name. The text is represented as a bag-of-words, with a feature space with dimensionality equal to 1836. The training set was obtained by sampling a subset of the entries (10%-50%), leaving the remaining ones for the test set. Like previous studies in the literature on this dataset (see e.g. [4]), the F1 score was employed to evaluate the tagger performances.

A knowledge base containing a set of 106 rules, expressed by FOL, has been created to express semantic relationships between the categories (see Table 1). The experiments tested the prediction capabilities of the classifiers, when considering all tags or only the 25 most popular tags in the dataset as output categories. In this second case, only the 8 logic rules fully defined over the subset of tags have been exploited.

[2] The dataset can be freely downloaded at: http://mulan.sourceforge.net/datasets.html

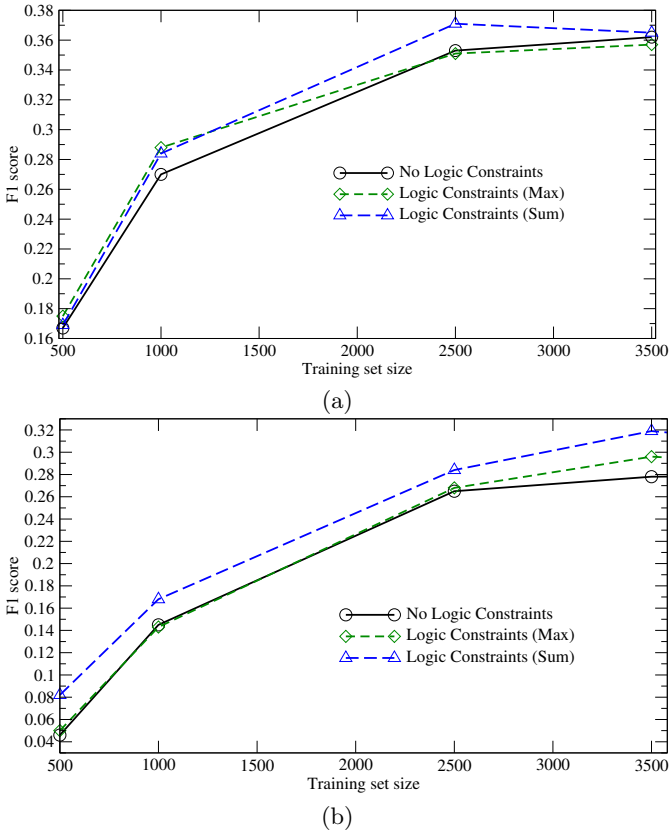
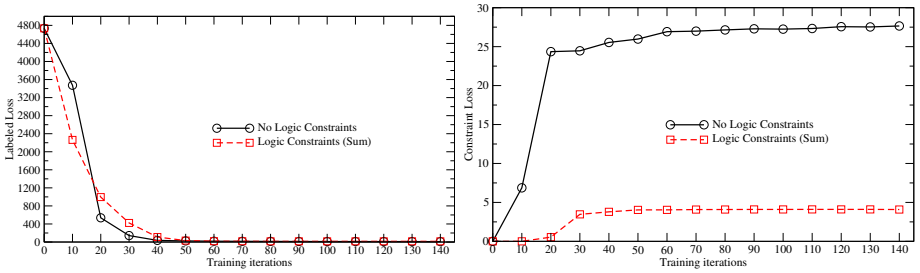



Fig. 1: F1 scores considering (a) the top 25 and (b) all the tags on the test set, when using or not using the knowledge base.

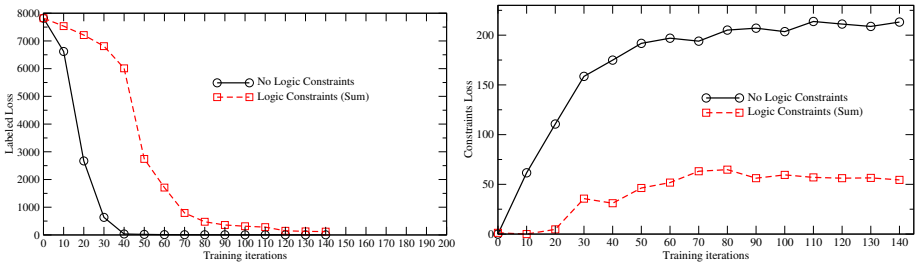
The rules correlate the tags and, after their conversion into the continuous form, they have been used to train the kernel machines according to the procedure described in the previous sections. The same experiments have been performed using a kernel machine trained using only the supervised examples. All the kernel machines used in this experiment have been based on a linear kernel, as they have been assessed as the best performers on this dataset in a first round of experiments (not reported in this paper).

Figure 1 reports the F1 scores computed over the test set provided by the 25 and 159 tag predictors, respectively. The logic constraints implemented via the mixture-by-sum, overperformed the logic constraints implemented as a mixture-by-max. Enforcing the logic constraints during learning was greatly beneficial with respect to a standard text classifier based on a kernel machine, as the F1 scores are improved by 2-4% when all tags are considered. This gain is consistent for the training set sizes that have been tested. When restricting the attention to




25 tags - Loss for labeled data

25 tags - Constraint error




100 tags - Loss for labeled data

100 tags - Constraint error

Fig. 2: Loss term on labeled data and on constraints deriving from the rules over the test set (generalization) for the 25 and all tag datasets.

the top 25 tags, the gains obtained by enforcing the mixture-by-sum constraints are smaller, ranging around 0-2%. These smaller gains are due to the significantly smaller number of logic rules that are defined over the 25 tags.

Figures 2 plots the loss on the labeled data and on the constraints for the test set (generalization) at the different iterations of the training for the 25 and 159 tag classifiers. In all the graphs, the loss on the constraint is low at the beginning of the training. This happens because all the provided rules are in the form of implications, which are trivially verified if the precondition is not true. Since no tags are yet assigned at the beginning of the training, the precondition of the rules is false, making the constraint verified. The figures show how the introduction of the constraints does not change significantly the loss on the labeled data at convergence, whereas the constraint loss is strongly reduced at the end of the training process. This means that the final weights of the kernel machine implement tagging functions that are able to fit much better the prior knowledge on the task.

## 5 Conclusions and Future Work

This paper presented a novel approach to text tagging, bridging pure machine learning approaches to knowledge based annotators based on logic formalisms.

The approach is based on directly injecting logic knowledge compiled as continuous constraints into the kernel machine learning algorithm. While previous work concentrated on t-norms, this paper presents a new approach to convert FOL clauses into constraints using mixtures of Gaussians. The experimental results show how the approach can over-perform a text annotator based on classical kernel machines by a significant margin. In this paper, we assumed that a logic predicate corresponds to each tag and, as a consequence, to a function to estimate. Therefore, the logic rules are defined directly over the tags. As future work we plan to test this approach to the case where new logic predicates can be added to the knowledge base. For example, ontology-based taggers use regular expressions to define partial tagging rules [6]. In our approach this would be implemented by assigning a new FOL predicate to each regular expression. After converting the logic rules as explained in this paper, it would be possible to train the kernel machines to learn jointly from examples and the KB.

## References


1. Bengio, S., Weston, J., Grangier, D.: Label embedding trees for large multi-class tasks. Advances in Neural Information Processing Systems 23, 163–171 (2010)
2. Caponnetto, A., Micchelli, C., Pontil, M., Ying, Y.: Universal Kernels for Multi-Task Learning. Journal of Machine Learning Research (2008)
3. Diligenti, M., Gori, M., Maggini, M., Rigutini, L.: Multitask Kernel-based Learning with Logic Constraints. In: Proceedings of the 19th European Conference on Artificial Intelligence. pp. 433–438. IOS Press (2010)
4. Katakis, I., Tsoumakas, G., Vlahavas, I.: Multilabel text classification for automated tag suggestion. ECML PKDD Discovery Challenge 75 (2008)
5. Klement, E., Mesiar, R., Pap, E.: Triangular Norms. Kluwer Academic Publisher (2000)
6. Laclavik, M., Seleng, M., Gatial, E., Balogh, Z., Hluchy, L.: Ontology based text annotation. In: Proceedings of the 18-th International Conference on Information Modelling and Knowledge Bases. pp. 311–315. IOS Press (2007)
7. Liu, D., Hua, X., Yang, L., Wang, M., Zhang, H.: Tag ranking. In: Proceedings of the 18th international conference on World wide web. pp. 351–360. ACM (2009)
8. Matusiak, K.: Towards user-centered indexing in digital image collections. OCLC Systems & Services 22(4), 283–298 (2006)
9. Peters, S., Denoyer, L., Gallinari, P.: Iterative Annotation of Multi-relational Social Networks. In: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining. pp. 96–103. IEEE (2010)
10. Sebastiani, F.: Machine learning in automated text categorization. ACM computing surveys (CSUR) 34(1), 1–47 (2002)
11. Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research 10, 207–244 (2009)
12. Zavitsanos, E., Tsatsaronis, G., Varlamis, I., Paliouras, G.: Scalable Semantic Annotation of Text Using Lexical and Web Resources. Artificial Intelligence: Theories, Models and Applications pp. 287–296 (2010)