Artificial Intelligence 204 (2013) 30–55

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Type Extension Trees for feature construction and learning in relational domains

Manfred Jaeger [a,*], Marco Lippi [b], Andrea Passerini [c], Paolo Frasconi [d]

[a] Institut for Datalogi, Aalborg Universitet, Denmark
[b] Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Università degli Studi di Siena, Italy
[c] Dipartimento di Ingegneria e Scienza dell'Informazione, Università degli Studi di Trento, Italy
[d] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Firenze, Italy

ARTICLE INFO

Article history:
Received 23 October 2012
Received in revised form 31 July 2013
Accepted 8 August 2013
Available online 20 August 2013

Keywords:
Statistical relational learning
Inductive logic programming
Feature discovery

ABSTRACT

Type Extension Trees are a powerful representation language for "count-of-count" features characterizing the combinatorial structure of neighborhoods of entities in relational domains. In this paper we present a learning algorithm for Type Extension Trees (TET) that discovers informative count-of-count features in the supervised learning setting. Experiments on bibliographic data show that TET-learning is able to discover the count-of-count feature underlying the definition of the h-index, and the inverse document frequency feature commonly used in information retrieval. We also introduce a metric on TET feature values. This metric is defined as a recursive application of the Wasserstein–Kantorovich metric. Experiments with a k-NN classifier show that exploiting the recursive count-of-count statistics encoded in TET values improves classification accuracy over alternative methods based on simple count statistics.

1. Introduction

Probabilistic logical (or relational) models provide models for properties and relationships of entities in domains with a relational structure, such as graphs, networks, or, generally, any kind of structure found in a relational database. The prevalence of this type of structured data, and the challenges posed by it for traditional machine learning methods based on simple attribute-value data models has led to an increasing interest over the past 10 years in probabilistic logical models, and associated statistical-relational learning techniques [10,6].

When modeling entities embedded in a relational domain a key question is what *features* of the entities are relevant to model and predict properties of interest. Apart from using attributes of the given entities themselves, one has in relational learning the ability to construct new features by considering the relational neighborhood of an entity. Taking into consideration related entities and their attributes, one obtains a basically unlimited supply of potential features.

A word on terminology here may be in order: by an *attribute* we mean a formal representation in a dataset of a property of individual entities by a data column. The color property of a flower, for example, could be formalized by attributes such as $color \in \{red, green, blue, orange, \ldots\}$, or three distinct attributes $RGB\_red, RGB\_green, RGB\_blue \in \{0, \ldots, 255\}$. The value space of an attribute will typically be a simple data type like Boolean, enumeration, or numeric. In classic attribute-value data, *feature* is often a synonym for attribute. By contrast, we use feature to denote formalized properties in a much broader sense. First, a feature may only be implicit in the data as a function of explicit data. For example, *brightness* as a function of RGB_red, RGB_green, and RGB_blue is a feature for an attribute-value dataset containing the attributes

* Corresponding author.
E-mail addresses: jaeger@cs.aau.dk (M. Jaeger), lippi@diism.unisi.it (M. Lippi), passerini@disi.unitn.it (A. Passerini), paolo.frasconi@unifi.it (P. Frasconi).

0004-3702/$ – see front matter
http://dx.doi.org/10.1016/j.artint.2013.08.002

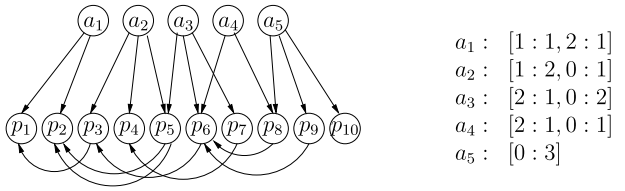


Fig. 1. Bibliographic data fragment and count-of-count feature.

RGB_red, *RGB_green*, *RGB_blue*; the "number of friends older than 22" is a feature (of a person entity) in a relational social-network dataset. Second, for relational data, a feature can also represent a property relating multiple entities. Thus, "titles having more than 3 words in common" would be a feature for a pair of paper entities in a bibliographic database. Third, unlike other common frameworks in statistical learning (like for example kernel methods), in this paper we are not interested in simple numerical features but will focus on features whose values are complex combinatorial data structures representing what we loosely will call counts-of-counts. In all cases, however, we require that a feature has a well-defined formal specification and value space. The language available for the formal specifications defines the feature space.

Relational learning frameworks differ widely to what extent they are linked to a clearly defined feature space, and to what extent feature selection or feature construction is integrated into the model learning process. On the one hand, there are techniques that only require the availability of features of a simple data type. The features construction is not part of the learning framework, and usually requires an application-dependent data pre-processing [42]. *Propositionalization* approaches also maintain a strict separation between feature construction and learning, but specific frameworks and representation languages for feature specification are a crucial ingredient [22].

On the other extreme there are approaches in which feature construction and model learning are tightly integrated, and, in fact, the learned model essentially consists of a list of features represented in a formal specification language. To this category belong most frameworks that are based on predicate logic as the feature representation language [3,20,21]. In between, there are approaches where feature construction is an integral part of the learning process, but the exact feature space accessible to the learner is less clearly delimited [1,18].

A key component in the design of relational features is given by the tools that are available for constructing features from properties of entities that are related to the entity of interest by chains of one or several relations. Since the number of entities that are reached by such "slotchains" [8] varies from instance to instance, this feature construction usually involves a form of combination or *aggregation* of the properties of multiple related entities.

In non-probabilistic inductive logic programming approaches, such an aggregation is usually based purely on existential quantification, i.e., a feature only determines whether or not a related entity with certain attributes exists. So, for example, for an "author" entity in a bibliographic database one could define a Boolean feature saying whether there exists a paper citing a paper of this author. A number of frameworks that are more closely linked to relational databases [31,13] construct features based on aggregation operators. Here, it would be possible, for example, to construct a feature that represents the average count of citations that papers of an author have received, or a feature that represents the average price of items purchased by a customer. Recently approaches that define probability distributions over entire structures based on pure count features have become quite popular [36,43]. Here the probability of a relational structure (over a given domain of entities) is determined by the count of entity tuples that satisfy some relational constraints, typically expressed as a logical clause.

All these approaches are based on features that only represent relatively simple summary statistics about quantitative properties of an entity's relational neighborhood. However, for many prediction tasks, a more detailed picture of combinatorial *count-of-count* features may be relevant. Consider the tiny bibliographic dataset shown in Fig. 1, for instance. It represents 5 different authors, 10 different papers by these authors, and citation links between the papers. Simple summary features for an author a could be the number of a's papers, or his/her total or average citation count. However, a currently important attribute for an author is the h-index [14]. To predict the h-index (or another attribute closely linked to the h-index – like receiving a professional award, or a large research grant) one may need to consider the more detailed feature of the count of papers with given counts of citations. In Fig. 1, the values for the 5 authors of this count-of-count feature are shown on the right (an expression $k : l$ meaning that there are l papers with k citations).

As another example for count of count features consider the Internet Movie Database (IMDB), a quite popular object of investigation in relational machine learning. Here one may be interested in predicting some attribute of a movie, e.g., whether it will be a box-office success (e.g. [38,31]). For this prediction one may consider the cast of the movie, for example in terms of its size, the count of actors in the cast who have previously received an award nomination, the total number of award nominations shared by the actors, etc. Again, a more detailed count-of-count feature can be more informative than only flat counts: it will make a difference, perhaps, whether there is a single actor in the cast with many award nominations (perhaps a single box office draw actor, but maybe beyond the peak of his/her career?), or whether there are many actors with one or two nominations each (perhaps a young all-star cast?).

In information retrieval, relevance measures for a document d given a query q are often based on counting terms appearing both in d and q. These counts will usually be weighted by a term weight such as *inverse document frequency*, which is given by the number of documents in the collection containing the term. Thus, the relevance measure is computed

from a count-of-count feature. Similarly to Fig. 1, for example, a query-document pair (d, q) could have a feature value $[3:1, 10:2, 7:1]$ expressing the fact that d and q have 1 term in common that appears in a total of 3 documents, 2 terms in common that each appear in 10 documents, and 1 term in common that appears in 7 documents.

Finally, consider the relational domain consisting of Web-pages and the *links-to* relation. An important attribute of a web-page is its pagerank [5], and we may want to estimate the pagerank of a web-page based on information of its local relational neighborhood. Unlike the h-index in the bibliographic example, which is precisely determined by a relational neighborhood of radius 2 defined by the chain $authorOf(A, P), cites(P', P)$, the pagerank[1] is fully determined only by the structure of the whole relational domain. However a useful approximation might be obtained already from local information. Clearly relevant for the pagerank of page P is the number of its incoming links. Also important is the pagerank of the pages P' linking to P, and hence the number of their incoming links. Furthermore, it is important to know for the pages P' linking to P the number of outgoing links of P' (pointing to pages other than P), because this determines how much of the pagerank of P' is "inherited" by P. Again, the full relevant information is only given by a comprehensive count-of-count feature.

The purpose of this paper is to develop a framework for the representation of rich combinatorial count-of-count features in the context of relational learning. The methodological contribution of the paper has three main components:

(C1) The definition of *Type-Extension Trees* (*TETs*) [7] as a formal representation language for count-of-count features.
(C2) A method for learning Type-Extension Trees in conjunction with a simple predictive model based on TET-features. This gives us a method for *relational feature discovery*, as well as a baseline supervised learning framework. In [7] TET learning employed the classic notion of information gain, while in this paper we take advantage of the recently introduced *relational information gain* [26].
(C3) The definition of a novel metric on TET feature values, which enables distance-based learning techniques that make use of count-of-count features in a substantial manner.

To illustrate the relationship and significance of these three components, consider an analogy with learning from standard numerical attribute-value data, where each instance is fully characterized by a tuple of numeric attributes, and a class variable. Fig. 2 on the left shows a small dataset with numeric attributes $A_1, A_2, N_1, \ldots, N_4$ and a binary class label with values p(ositive) and n(egative). In this dataset the class is only correlated with the attributes A_1 and A_2, whereas $N_1, \ldots, N_4$ are random noise. The plot in the left part of the figure represents the values of A_1, A_2 and C. The relevant feature subset for predicting the class label then is $\{A_1, A_2\}$, out of the space of all possible feature subsets (Fig. 2**A**). The set $\{A_1, A_2\}$ may also be called a *sufficient* or *model-independent* feature for predicting C. No concrete type of machine learning model will use all the information represented by this feature. A decision tree model, for instance, will only use finitely many Boolean features defined by lower- or upper-bounds on A_1, A_2-values. A linear classifier will only use a linear function of A_1, A_2. We may call the spaces of such features *reduced* or *model-specific* feature spaces (Fig. 2**B**). A proper distinction between the sufficient and reduced feature spaces is important when we interpret the result of learning a specific model also in terms of feature discovery: the decision tree learned from our example dataset (Fig. 2**C** left) uses four Boolean features $A_1 > 0.79, \ldots, A_2 > 0.75$, and, strictly speaking, has "discovered" exactly these four features. However, one will typically want to generalize, take $\{A_1, A_2\}$ as the discovered feature in the sufficient feature space, and assume that suitable reductions of $\{A_1, A_2\}$ in other model-specific feature spaces will also lead to good performance of other types of models. Fitting a logistic regression model to our data (Fig. 2**C** right) directly leads only to the construction of the linear function $12.8 - 22.1A_1 \cdots + 3.4N_4$ as a predictive feature. Again, one can abstract from this reduced feature, and try to identify the "discovered" model-independent feature. Here this abstraction is not as clear-cut as in the decision tree case. Considering the attributes whose coefficients in the linear function have the highest absolute values for inclusion in the feature subset, one here might take any of the subsets $\{A_1\}$, $\{A_1, A_2\}$, $\{A_1, A_2, N_3, N_4\}$ as the discovered model-independent feature.

Our objective (C1) aims at defining for relational data a rich, model-independent feature space that corresponds to the space of attribute subsets (Fig. 2**A**), and that includes complex count-of-count features. This definition is developed in Section 2 via syntax and semantics of Type Extension Trees (see Section 2.1). The construction of a sufficient feature space for relational data not only faces the challenge of the basically unlimited supply of possible features, but also the challenge of the diversity of relational learning tasks: attribute prediction for individual entities, link prediction, and classification of whole relational structures (e.g. molecules) require the specification not only of features for single entities, but also for pairs of entities, tuples of entities, and global features of a whole relational dataset. TET features provide a uniform and coherent framework for all these cases.

Component (C2) corresponds to the feature discovery process via the learning of a lightweight model as illustrated in Fig. 2. This is achieved by first defining directly on TET-feature values a *discriminant function* that turns a TET-feature into a predictive model (Section 3). Based on this discriminant function a TET structure learning algorithm is developed that, strictly speaking, will discover a feature (T, f) (T a TET, f a discriminant function defined on T), but from which in a trivial abstraction step we can extract the model-independent TET feature T (Section 5).

[1] In fact pagerank can be applied beyond Web searching, e.g. in bibliometrics [2] or collaborative filtering [27].

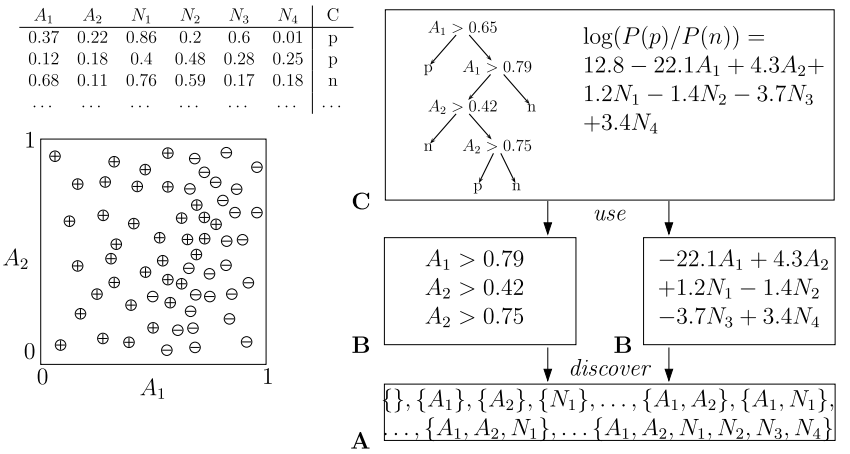


Fig. 2. Analogy: numerical data. **A**: Sufficient feature space, **B**: Model-specific feature spaces, **C**: "lightweight" learned models.

Given a TET – either learned or manually constructed based on expert knowledge – we finally in component (C3) define a predictive model based on TET features that makes use of the TET's count-of-count values in a more substantial and sophisticated manner than the discriminant function. For this we define a metric on TET values that then enables nearest neighbor classification (Section 4).

2. Feature representation with Type Extension Trees

2.1. TET syntax and semantics

In this section, we review the basic syntax and semantics definitions of Type Extension Trees [17]. To simplify the definitions, we will assume that all attributes and relations are Boolean, which means that a multi-valued attribute like 'color' that would give rise to atomic propositions such as color(hat, red) is assumed to be encoded using Boolean attributes like color_red(hat). Relational data can then be viewed as a model in the sense of the following definition.

Definition 2.1. Let R be a *relational signature*, i.e. a set of relation symbols of different arities. A (finite) model for R, $\mathcal{M} = (M, I)$ consists of a finite domain M, and an interpretation function $I : r(\boldsymbol{a}) \to \{t, f\}$ defined for all *ground atoms* $r(\boldsymbol{a})$ constructible from relations $r \in R$ and arguments $\boldsymbol{a} \in M^{arity(r)}$.

Throughout the paper we use f and t as shorthands for false and true. Furthermore, we denote objects (entities) from a domain with lowercase letters, and logical variables with uppercase letters. Only first-order formulas are acceptable, i.e. variables always stand for objects. Bold symbols always denote tuples of the corresponding non-bold symbols, e.g., in the foregoing definition: $\boldsymbol{a} = (a_1, \ldots, a_{arity(r)})$. In logic programming terminology, $\mathcal{M}$ is a Herbrand interpretation for the signature consisting of R and constant symbols for the elements of M. For convenience we may assume that the domain M is partitioned into objects of different *types*, that arguments of relations are typed, and that I is only defined for ground atoms with arguments of appropriate types. For the sake of simplicity we do not introduce any special notation for specifying types. Rather, in our examples below we will typically use generic capital letters (X, Y, Z, U, V, W) to indicate variables which may range over all domain objects, and specific letters (like A for author or P for paper) to implicitly mean that the corresponding variables are restricted to a subset of objects. With a slight abuse of notation, if $\tau(\boldsymbol{a})$ is a complex ground sentence, we denote by $I(\tau(\boldsymbol{a}))$ its truth value under interpretation I.

Definition 2.2. An *R-literal* is a (negated) atom $r(\boldsymbol{V})$ ($r \in R \cup \{=\}$, $\boldsymbol{V}$ a tuple of variable symbols). We also allow the special literal $\top(\boldsymbol{V})$, which always evaluates to t. An *R-type* is a conjunction of R-literals.

A *type extension tree* (TET) over R is a tree whose nodes are labeled with R-types, and whose edges are labeled with (possibly empty) sets of variables.

In the following we will usually omit the reference to the underlying signature R, and talk about literals and types, rather than R-literals and R-types.

Note that according to Definition 2.2 a literal cannot contain any constant symbols as arguments. Since R is assumed to only contain relation and no constant symbols, this is consistent with the usual definition of an R-literal. The term "type" for a conjunction of literals is motivated by two distinct (yet compatible), existing uses of "type": on the one hand, the type of an entity is commonly understood as a property expressed by a single unary predicate, e.g., *movie*(V) or *person*(V). Our definition generalizes this to types of tuples of objects, and to types which are expressed via a conjunction of literals. On the

other hand, "type" is used in mathematical model theory for consistent sets of formulas using free variables $V_1, \ldots, V_n$ that describe properties of n-tuples of domain elements [15]. Our definition is a special case of type in this sense by limiting it to a single, quantifier-free conjunction of literals.

Example 2.3. The following is a TET for a signature containing relations *author*, *authorOf* and *cites*:

$$author(A) \xrightarrow{P_1} authorOf(A, P_1) \xrightarrow{P_2} cites(P_2, P_1). \tag{1}$$

According to the semantics given below, this TET represents a feature that is sufficient for computing the h-index of an author, for example.

Example 2.4. A TET that will be sufficient for representing relevance features based on inverse-document frequency weights is

$$\begin{array}{c} \top(D, Q) \\ \Big\downarrow T \\ term\_in\_document(T, D), term\_in\_query(T, Q) \\ \Big\downarrow D' \\ term\_in\_document\big(T, D'\big) \end{array} \tag{2}$$

Labeled edges in a TET are related to quantifiers in predicate logic: like a quantifier, a labeled edge *binds* all occurrences of the variables associated with the edge in the subtree rooted at this edge. The *free variables* of a TET are all variables not bound by an edge label. We call a TET *propositional* if all edge labels are empty. The TET in (1) has the single free variable A, the one in (2) the two free variables D and Q. In both cases, the root node essentially serves to introduce the free variables, and, in case of (1), to explicitly establish that the variable ranges over entities of type author. If such type constraints on variables are assumed to be implicit in the variable names, then the root will usually be a vacuous $\top()$ atom, as in (2).

We write $T(\boldsymbol{V})$ to denote a TET whose free variables are among the variables $\boldsymbol{V}$ (but does not necessarily contain all of them). We write

$$T(\boldsymbol{V}) = \big[\tau(\boldsymbol{V}), \big(\boldsymbol{W}_1, T_1(\boldsymbol{V}, \boldsymbol{W}_1)\big), \ldots, \big(\boldsymbol{W}_m, T_m(\boldsymbol{V}, \boldsymbol{W}_m)\big)\big] \tag{3}$$

to denote a TET with a root labeled with $\tau(\boldsymbol{V})$, and m sub-trees $T_1(\boldsymbol{V}, \boldsymbol{W}_i,)$ reached by edges labeled with variables $\boldsymbol{W}_i$ (possibly empty).

A TET $T(\boldsymbol{V})$ with free variables $\boldsymbol{V} = V_1, \ldots, V_k$ will define a *feature* for k-tuples of domain entities: for any model $\mathcal{M}$, and any $\boldsymbol{a} \in M^k$ the TET defines a *feature value* $V(T(\boldsymbol{a}))$. Fig. 1 on the right shows (in a somewhat simplified form) the values $V(T(a_1)), \ldots, V(T(a_5))$ for the TET $T(A)$ in (1). We give the general definition of TET semantics in two steps: first we define the value space of nested counts associated with a given TET $T(\boldsymbol{V})$, and then the actual mapping $\boldsymbol{a} \mapsto V(T(\boldsymbol{a}))$.

Definition 2.5. For any set A we denote with $multisets(A)$ the set of all multisets over A. We denote with $\{a_1 : k_1, \ldots, a_n : k_n\}$ a multiset that contains k_i copies of a_i. The *value space* $\mathcal{V}(T)$ of a TET T is inductively defined as follows:

Base: If $T = [\tau]$ consists of a single node, then $\mathcal{V}(T) = \{t, f\}$.

Induction: If $T = [\tau, (\boldsymbol{W}_1, T_1), \ldots, (\boldsymbol{W}_m, T_m)]$, then

$$\mathcal{V}(T) = \{f\} \cup \{t\} \times \bigtimes_{i=1}^{m} multisets\big(\mathcal{V}(T_i)\big).$$

We note that according to this definition the structure of $\mathcal{V}(T)$ only depends on the tree structure of T, but not on the labeling of the edges of T, or the types at the nodes of T.

Example 2.6. (1) is a graphical representation of a TET that following (3) can be written as

$$T(A) = \big[author(A), \big(P_1, \big[authorOf(A, P_1), \big(P_2, \big[cites(P_2, P_1)\big]\big)\big]\big)\big]. \tag{4}$$

The recursive definition of $\mathcal{V}(T)$ is grounded in $\mathcal{V}([cites(P_2, P_1)]) = \{t, f\}$. In other words, the single node TET $[cites(P_2, P_1)]$ represents a Boolean feature for pairs of papers. The inductive construction proceeds with the definition of $\mathcal{V}(T'(A, P_1))$, where

$$T'(A, P_1) = \big[authorOf(A, P_1), \big(P_2, \big[cites(P_2, P_1)\big]\big)\big] \tag{5}$$

represents a feature of an author-paper pair A, P_1. This value space is constructed according to the inductive case of Definition 2.5 as the union of $\{f\}$ and pairs of the form (t, A), where A is a multiset of t, f values. Thus, examples are: f

(according to Definition 2.7 below, this is the feature value of an author-paper pair (a, p), where a is not the author of p), $(t, \{f : 9, t : 1\})$ (this will be the feature value, e.g., of the pairs (a_1, p_1) and (a_2, p_4) in Fig. 1), or $(t, \{f : 8, t : 2\})$ (the feature value of (a_1, p_2) in Fig. 1).

Finally, values of the full TET (4) are either f, or (t, A), where A is a multiset of values from $\mathcal{V}(T'(A, P_1))$. Examples are

$$\begin{aligned}
\gamma_1\colon &\quad \left(t, \left\{\begin{array}{l} f:8 \\ (t,\{f:9,t:1\}):1 \\ (t,\{f:8,t:2\}):1 \end{array}\right\}\right) \\
\gamma_2\colon &\quad \left(t, \left\{\begin{array}{l} f:7 \\ (t,\{f:9,t:1\}):2 \\ (t,\{f:10\}):1 \end{array}\right\}\right) \\
\gamma_3\colon &\quad \left(t, \left\{\begin{array}{l} f:7 \\ (t,\{f:8,t:2\}):1 \\ (t,\{f:10\}):2 \end{array}\right\}\right) \\
\gamma_4\colon &\quad \left(t, \left\{\begin{array}{l} f:8 \\ (t,\{f:8,t:2\}):1 \\ (t,\{f:10\}):1 \end{array}\right\}\right) \\
\gamma_5\colon &\quad \left(t, \left\{\begin{array}{l} f:7 \\ (t,\{f:10\}):3 \end{array}\right\}\right).
\end{aligned} \tag{6}$$

Here, for better readability, the outer multisets are written as column vectors, rather than in comma-separated linear form (used for the inner multisets). These five values are just the feature values of the five authors $a_1, \ldots, a_5$ in Fig. 1. We will usually use γ to denote TET values. We note that the t component of a value of the form (t, A), A a multiset, is redundant: since every occurrence of a multiset is prefixed by such a t one could just write A instead of (t, A). However, adding the explicit t as an embellishment to the value lets us maintain a clearer match between the structure of a TET T and its values $\gamma \in \mathcal{V}(T)$: otherwise, for example, the value $(t, \{f : 1\})$ of a two-level TET would become just $\{f : 1\}$ and easily confused with a value f.

In the preceding example we have already introduced the values of the TET feature (1) for the entities $a_1, \ldots, a_5$ in Fig. 1. These values refine the informal count-of-counts shown in Fig. 1 by representing in a more principled way the recursive nature of count-of-counts, and by also including f counts (in this case, the number of papers not written by a given author, and the number of papers not citing a given paper). In the following we give the general definition of the feature value $V(T(\boldsymbol{a})) \in \mathcal{V}(T(\boldsymbol{V}))$ for a specific tuple $\boldsymbol{a}$.

Definition 2.7. Let $\mathcal{M} = (M, I)$ be a model, $T(V_1, \ldots, V_k)$ a TET, and $\boldsymbol{a} \in M^k$. The value $V(T(\boldsymbol{a})) \in \mathcal{V}(T)$ is defined as follows:

Base: If $T(\boldsymbol{V}) = [\tau(\boldsymbol{V})]$ consists of a single node, then $V(T(\boldsymbol{a})) := I(\tau(\boldsymbol{a}))$.

Induction: If $T(\boldsymbol{V}) = [\tau(\boldsymbol{V}), (\boldsymbol{W}_1, T_1(\boldsymbol{V}, \boldsymbol{W}_1)), \ldots, (\boldsymbol{W}_m, T_m(\boldsymbol{V}, \boldsymbol{W}_m))]$:

(a) If $I(\tau(\boldsymbol{a})) = f$ then $V(T(\boldsymbol{a})) := f$.
(b) If $I(\tau(\boldsymbol{a})) = t$ then

$$V\big(T(\boldsymbol{a})\big) := \big(t,\ \mu(\boldsymbol{a}, \boldsymbol{W}_1, T_1), \ldots, \mu(\boldsymbol{a}, \boldsymbol{W}_m, T_m)\big),$$

with $\mu(\boldsymbol{a}, \boldsymbol{W}_i, T_i) \in \mathit{multisets}(\mathcal{V}(T_i))$ given by

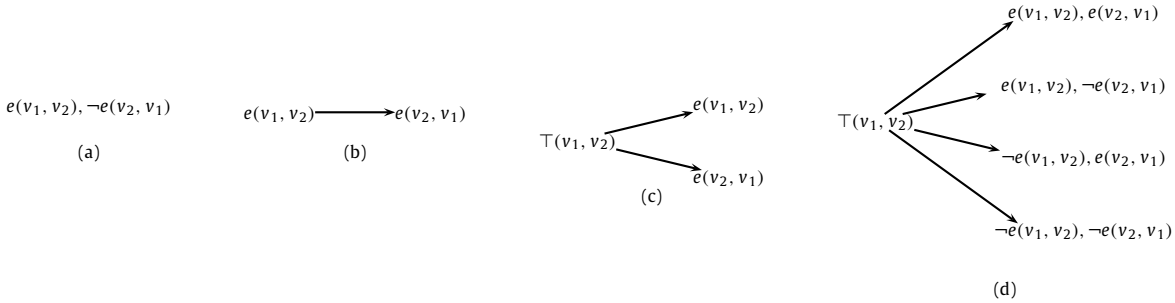
$$\big\{\gamma : \big|\big\{\boldsymbol{b} \in M^{k_i} \text{ s.t. } V\big(T_i(\boldsymbol{a}, \boldsymbol{b})\big) = \gamma\big\}\big|\big\}, \tag{7}$$

where γ ranges over all values in $\mathcal{V}(T_i)$, and k_i is the number of variables in $\boldsymbol{W}_i$.

We remark that the original definitions given in [17] treated the cases $I(\tau(\boldsymbol{a})) = f$ and $I(\tau(\boldsymbol{a})) = t$ symmetrically, which is why values according to Definition 2.5 were called "pruned" in [7] (because here the recursive evaluation of counts is cut off once a node evaluating to f is encountered on a TET branch).

We next consider in some detail the special case of unlabeled edges in a TET, and, in particular, purely propositional TETs. Consider a TET with a single unlabeled branch attached to the root: $T = [\tau(\boldsymbol{V}), (\emptyset, T_1(\boldsymbol{V}))]$. The multiset (7) then contains the single value $\gamma = V(T_1(\boldsymbol{a}))$. Generally, an unlabeled edge in a TET induces in the recursive structure of $\mathcal{V}(T)$ a single value of the sub-TET T' reached by this edge, whereas an edge labeled with one or several variables induces a multiset of such values. The following example serves to illustrate the nature of propositional TETs. At the same time the example shows how different tree structures of a TET can be used to represent different logical properties.

Table 1
Propositional TETs and their values.



	(a)	(b)	(c)	(d)
$\overset{a_1}{\bullet}\ \overset{a_2}{\bullet}$	f	f	$(t, \{f\}, \{f\})$	$(t, \{f\}, \{f\}, \{f\}, \{t\})$
$\overset{a_1}{\bullet}\rightarrow\overset{a_2}{\bullet}$	t	$(t, \{f\})$	$(t, \{t\}, \{f\})$	$(t, \{f\}, \{t\}, \{f\}, \{f\})$
$\overset{a_1}{\bullet}\leftarrow\overset{a_2}{\bullet}$	f	f	$(t, \{f\}, \{t\})$	$(t, \{f\}, \{f\}, \{t\}, \{f\})$
$\overset{a_1}{\bullet}\leftrightarrow\overset{a_2}{\bullet}$	f	$(t, \{t\})$	$(t, \{t\}, \{t\})$	$(t, \{t\}, \{f\}, \{f\}, \{f\})$

Example 2.8. Consider a propositional TET $T(\boldsymbol{V})$ with $\boldsymbol{V} = v_1, \ldots, v_n$. Since no further variables are introduced via edge labels, then all nodes only contain literals in the variables $\boldsymbol{V}$, and therefore $V(T(a_1, \ldots, a_n))$ only depends on the relational sub-structure induced by $a_1, \ldots, a_n$ in $\mathcal{M}$. By varying the types at the nodes, as well as the tree structure of $T(\boldsymbol{V})$, the values $V(T(\boldsymbol{a}))$ can represent a variety of different structural properties.

Consider a relational signature that contains a single binary ("edge") relation symbol $e(\cdot,\cdot)$. The upper part of Table 1 shows 4 different propositional TETs with the two free variables v_1, v_2. In all cases, the TET value $T(a_1, a_2)$ is determined by the relational sub-structure induced by a_1, a_2.

The leftmost column in Table 1 lists the four different possible sub-structures, and the remaining columns the values returned by TETs (a)–(d) for these sub-structures.

TET (a) represents a feature that only tests whether a_1, a_2 define the sub-structure $\overset{a_1}{\bullet}\rightarrow\overset{a_2}{\bullet}$. TET (b) is doing a two-stage test for the two possible edge relations. If the first test fails, i.e. there is no edge $e(a_1, a_2)$ then the value is *false*, regardless of the presence or absence of the converse edge.

Both (c) and (d) use a vacuous root-type $\top(v_1, v_2)$ to connect sub-TETs that are evaluated separately. The values in columns (c) and (d) in the table list the values of the different sub-TETs according to the top-to-bottom order on the branches defined by the graphical representation above. Both TETs discriminate between all four a_1, a_2-structures in the sense that each structure has a unique value. In this sense, (d) could be seen as a redundant version of (c). However, as we will see in Section 4, (c) and (d) exhibit quite distinct behavior with respect to the metric we will define on TET values.

2.2. TET definable features

In this section we illustrate that TETs provide a representation language for a rich class of fundamental features that are usable in a variety of learning frameworks, and that, thus, TET-features can play the role of the sufficient feature space in analogy to Fig. 2**A**.

Example 2.9. *Inductive Logic Programming* (*ILP*) is probably the oldest approach to relational learning. In ILP one learns classification rules for a target predicate, usually in the form of logical clauses. A classic example for ILP are *Bongard* problems, where scenes consisting of geometric objects have to be classified as positive or negative examples. Here, any such scene is a relational model, whose domain consists of geometric objects on which attributes like *triangle*(X) and *in*(X, Y) are defined. Then a rule for the *positive* scene examples can be

$$positive \leftarrow circle(X) \wedge triangle(Y) \wedge in(Y, X),$$

which says that a scene is positive, if it contains a triangle inside a circle (cf. [35, Chapter 4]). The feature used by this classification rule, thus, is a reduction of the TET feature

$$\top \xrightarrow{X,Y} circle(X) \wedge triangle(Y) \wedge in(Y, X).$$

Values of this simple TET are of the form $(t, \{f : k, t : l\})$, giving the counts of object pairs that satisfy, respectively do not satisfy, the body of the rule. The reduced ILP feature is just the Boolean that tests whether $l > 0$.

The original limitation to Boolean features defined by existential quantification has in some ILP based approaches been lifted via the introduction of special literals that contain aggregators for quantitative information. For example, [44] introduce *aggregate conditions*, which now would allow for a rule like

$$positive \leftarrow circle(X) \wedge \text{count}\{Y \mid triangle(Y) \wedge in(Y, X)\} \geqslant 3,$$

which says that a scene is positive, if it contains a circle with at least three triangles inside. The Boolean feature used by this rule is a reduction of the TET feature

$$\top \xrightarrow{X} circle(X) \xrightarrow{Y} triangle(Y) \wedge in(Y, X). \tag{8}$$

While not explicitly explored in [44], nested aggregate conditions also could be possible:

$$positive \leftarrow count\{X \mid circle(X), \text{count}\{Y \mid triangle(Y) \wedge in(Y, X)\} \geqslant 3\} \geqslant 2$$

would now classify a scene as positive if it contains at least two circles, each with at least three triangles inside. This Boolean feature still is a reduction of the underlying count-of-count feature represented by (8).

Several other frameworks have been proposed for constructing numeric or Boolean features by iterated aggregation over chains of relational dependencies. Some of these also are in the ILP tradition [22], others arise out of a combination of relational extensions of probabilistic graphical models or decision trees [8,19,1,31].

While in this previous work nested aggregation was considered, all these frameworks require an immediate aggregation at each step along a chain of relations. Thus, considering bibliographic data for example, one could define an author feature that represents whether the author has at least seven publications with at least seven citations each. However, a complex count-of-count feature that would be sufficient for computing the h-index is outside the scope of these feature construction methods.

Most of the approaches here mentioned also consider (or, in some cases, focus on) numeric predicates, and aggregation of numeric values using functions like *mean* and *max*. For the purely categorical (especially Boolean) data that we consider in this paper, the only common aggregation function is *count* (with "*exists*" as the special case "*count* $\geqslant 1$"). In spite of our current restriction to Boolean data, the same basic TET architecture could also be extended to numeric data, and thereby also be used to represent the underlying sufficient combinatorial and numerical information needed to compute more model-specific aggregated features.

Example 2.10. In this example we focus on *Markov-Logic Networks* (*MLNs*) [36] in the generative setting. In this case MLNs define a distribution over all models $\mathcal{M}$ for a given signature R, and a fixed domain M. This distribution is defined by a knowledge base KB containing first-order logic formulas ϕ_i with attached numeric weights w_i:

$$\text{KB:} \quad \begin{array}{ll} \phi_1(\boldsymbol{X}_1) & w_1 \\ \ldots & \ldots \\ \phi_n(\boldsymbol{X}_n) & w_n \end{array} \tag{9}$$

We refer to [36] for the details of MLN syntax and semantics. Relevant in the current context is the fact that the distribution is defined as a function of count features, which, adapting the notation of the previous example, can be written in the form

$$count\{\boldsymbol{X} \mid \phi(\boldsymbol{X})\} \tag{10}$$

where now $\phi(\boldsymbol{X})$ can be any first-order formula with free variables $\boldsymbol{X}$. While, thus, very similar in appearance to the aggregate features of the preceding example, there are some essential differences: first, MLNs depend on the actual integer-valued count feature, not only on derived Boolean features of the form $count\{\ldots\} \geqslant k$. Second, (10) takes the count of all substitutions of tuples of domain elements for all the free variables $\boldsymbol{X}$ in $\phi(\boldsymbol{X})$. As a result, whereas $count\{Y \mid triangle(Y) \wedge in(Y, X)\} \geqslant 3$ was a feature of the object X, now $count\{\boldsymbol{X} \mid \phi(\boldsymbol{X})\}$ is a feature of entire models $\mathcal{M}$.

Concrete MLN implementations will often allow only a restricted class of formulas $\phi(\boldsymbol{X})$ in the model specification (e.g. quantifier free formulas). However, the use of arbitrary first-order formulas poses no problem at the level of semantic definitions, and so we here consider models with no restrictions on the $\phi(\boldsymbol{X})$. For any such count feature one can construct a TET $T()$ such that the integer-valued feature (10) is obtained as a reduction of the values $V(T())$. We construct $T()$ in the form $\top \xrightarrow{\boldsymbol{X}} T_\phi(\boldsymbol{X})$, where now $T_\phi(\boldsymbol{X})$ is a TET with free variables $\boldsymbol{X}$, such that the truth value of $\phi(\boldsymbol{a})$ can be read off the TET value $V(T_\phi(\boldsymbol{a}))$.

The construction of T_ϕ is by induction on the structure of ϕ. In the atomic case $\phi(\boldsymbol{X}) \equiv r(\boldsymbol{X})$ T_ϕ contains the single node $r(\boldsymbol{X})$. For a conjunction $\phi(\boldsymbol{X}) = \phi_1(\boldsymbol{X}_1) \wedge \phi_2(\boldsymbol{X}_2)$ one lets $T_\phi(\boldsymbol{X}) = [\top, (\emptyset, T_{\phi_1}(\boldsymbol{X}_1)), (\emptyset, T_{\phi_2}(\boldsymbol{X}_2))]$. At this point in the construction TET values $V(T_\phi(\boldsymbol{a}))$ already encode more information than the mere Boolean value $\phi(\boldsymbol{a})$, since it also contains the individual truth values for the two conjuncts ϕ_1, ϕ_2 (cf. (c) in Table 1). For the negation case $\phi(\boldsymbol{X}) \equiv \neg\psi(\boldsymbol{X})$ one can simply let $T_\phi = T_\psi$: if the truth value of $\psi(\boldsymbol{a})$ is retrievable from $V(T_\psi(\boldsymbol{a}))$, then so is the truth value of $\neg\psi(\boldsymbol{a})$. In the quantifier case $\phi(\boldsymbol{X}) \equiv \exists Y \psi(\boldsymbol{Z}, Y)$ $(\boldsymbol{X} = \boldsymbol{Z} \cup Y)$ one defines $T_\phi(\boldsymbol{X}) = [\top, (Y, T_\psi(\boldsymbol{Z}, Y))]$. Again, the TET constructed in this step represents a counting-refinement of the Boolean feature actually required (note that while (10) has an outermost counting semantics for the variables $\boldsymbol{X}$, there is a standard Boolean semantics for any quantifiers appearing internally in ϕ).

Example 2.11. Graphs are a special kind of relational models in which there is a single binary (edge) relation $e(X, Y)$, and, in the case of labeled graphs, multiple unary relations $l_i(X)$ representing the different node labels. An important feature

considered in graph mining problems, and used, for example, to define kernel functions on graphs [9], is the number of subgraphs of a specific structure: if H is any (finite) graph, then

$$\phi_H(G) := \left|\left\{H' \mid H' \subseteq G, H' \simeq H\right\}\right|$$

counts the number of subgraphs H' isomorphic to H that are embedded in G. The (unlabeled) graph H with nodes $\{1,\ldots,k\}$ can be described by a type that contains one variable X_i for each node i of H, and the literal $e(X_i, X_j)$ for each edge $i \to j$ in H. Then

$$\top \xrightarrow{X_1,\ldots,X_k} \bigwedge_{(i,j):e(i,j)} e(X_i, X_j)$$

is just the TET feature that corresponds to $\phi_H(G)$. The case of labeled graphs, or the case where $H' \subseteq G$ is the induced subgraph relationship (i.e., there are no edges in G between any nodes matched to nodes of H', other than the edges also present in H') are treated in a similar manner.

The preceding examples illustrate the ability of the TET language to represent in a coherent manner a wide range of features used in a variety of different relational learning frameworks. There are some limitations to the TET language, though: TETs are essentially rooted in first-order logic, with a refined counting semantics replacing Boolean existential or universal quantification. They cannot represent features that are not first-order in nature, such as features that are defined in terms of the transitive closure of a relation, which can only be defined in suitable extensions of first-order logic, such as least fixed-point logic, or transitive closure logic [11]. For example, "paper p_1 can be found by tracing a chain of citations starting with paper p_2" is not TET-expressible. Also, the integer-valued feature "Erdös number of author a" cannot be captured by a TET, since it depends on a chain of co-authorship relations of undetermined length.

3. TET discriminant function

A TET alone only defines a feature of objects in a relational domain. TET-defined features can be incorporated in many ways into existing types of predictive or descriptive models. For example, one can define distance or kernel functions on TET value spaces $\mathcal{V}(T)$, thereby making TET features usable for standard clustering techniques, or SVM classifiers. We will introduce a metric on $\mathcal{V}(T)$ in Section 4. In this section we first describe how to build a predictive model on a TET feature using simple *discriminant functions* on TET values, i.e. functions of the form [7]

$$d : \mathcal{V}(T) \to \mathbb{R}.$$

Such discriminant functions directly lead to binary classification models. We use $\{+,-\}$ to denote binary class labels. Then one can learn two discriminant functions d_+, d_-, and a threshold value t, and assign class label $+$ to a tuple $\boldsymbol{a}$ iff

$$d_+\big(V\big(T(\boldsymbol{a})\big)\big)/d_-\big(V\big(T(\boldsymbol{a})\big)\big) > t. \tag{11}$$

3.1. Defining the simple TET discriminant

We now introduce one simple type of TET-discriminant function. The motivation for the particular form of discriminant function we propose is twofold: first, for a given TET, these discriminants are efficient to learn and evaluate. Since the discriminant function is also used in TET learning, within a wrapper evaluation routine for candidate TETs (see Section 5), efficiency is an important issue. Second, as we will show below, our discriminant function definition can be motivated as a uniform generalization of the classic decision tree and Naive Bayes models.

Definition 3.1. Let T be a TET. A *weight assignment* $\boldsymbol{\beta}$ for T assigns a non-negative real number to all nodes of T. A weight assignment can be written as $(\beta^r, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m)$, where β^r is the weight assigned to the root, and $\boldsymbol{\beta}_i$ is the weight assignment to the ith sub-tree.

For a TET T with node weights $\boldsymbol{\beta}$ we define the discriminant function $d^{\boldsymbol{\beta}}$ as follows. Let $\gamma \in \mathcal{V}(T)$:

- If $\gamma = f$ define $d^{\boldsymbol{\beta}}(\gamma) := 0$.
- If $\gamma = t$ then $T = [\tau(\boldsymbol{V})]$ consists of a single node, and $\boldsymbol{\beta} = (\beta^r)$. Define $d^{\boldsymbol{\beta}}(\gamma) := \beta^r$.
- If $\gamma = (t, \mu_1, \ldots, \mu_m)$, $\mu_i \in \mathit{multisets}(\mathcal{V}(T_i))$, define

$$d^{\boldsymbol{\beta}}(\gamma) := \beta^r \cdot \prod_{i=1}^{m} \prod_{\gamma' \in \mu_i, \gamma' \neq f} \frac{1}{\beta^r} d^{\boldsymbol{\beta}_i}(\gamma').$$

In the following we illustrate the nature of this discriminant function. We begin by investigating TETs in a propositional setting, where it turns out that our simple discriminant is closely related to the standard decision tree and Naive Bayes models.

Table 2
A propositional dataset.

$b(X)$	$c(X)$	$a(X)$	
		t	f
t	t	1;15	10;6
t	f	0;10	9;3
f	t	5;3	15;9
f	f	4;2	6;2

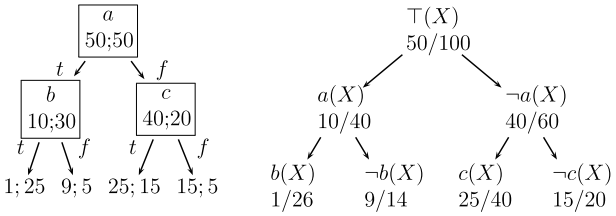


Fig. 3. Decision tree and TET discriminant d_+.

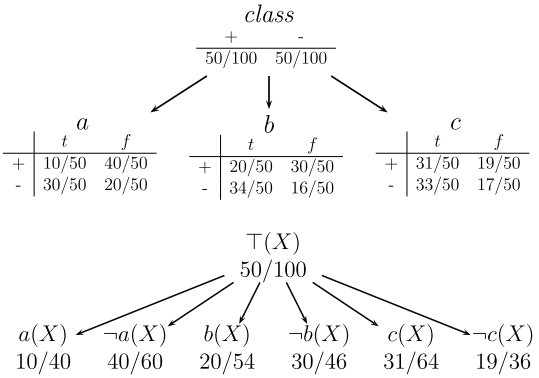


Fig. 4. Naive Bayes model and TET discriminant d_+.

Table 2 shows a hypothetical dataset with observations of 100 cases for which three Boolean attributes a, b, c and a binary class label *class* with values $+$, $-$ are recorded. To be consistent with our relational notation, we view the Boolean attributes as unary relations $a(X)$, $b(X)$, $c(X)$ defined on the observations X. The entries in the table represent pairs $n^+; n^-$ of counts for the positive and negative class with the given attribute value combination.

Fig. 3 on the left shows a decision tree that could be constructed for this data. The nodes are labeled with the counts $n^+; n^-$ of positive/negative examples that reach the nodes. An example e with attribute values $a(e) = t$, $b(e) = f$, $c(e) = t$, for instance, would be classified as positive, or, more precisely, would be estimated as being positive with probability 9/14.

The right side of the Fig. 3 shows a propositional TET for a, b, c. This TET is labeled with a weight assignment, where the weight at each node corresponds to the empirical frequency $n^+/(n^+ + n^-)$ of the positive class among the examples that satisfy all the conditions on the path from the root down to the node. For the example $a(e) = t$, $b(e) = f$, $c(e) = t$ the TET in Fig. 3 evaluates to

$$V\big(T(e)\big) = \big(t, \big\{\big(t, \{f\}, \{t\}\big)\big\}, \{f\}\big).$$

For this value, the discriminant function evaluates as

$$d_+^{\beta}\big(V\big(T(e)\big)\big) = 50/100\frac{10/40}{50/100}\frac{9/14}{10/40} = 9/14 = P\big(+ \,\big|\, a(e) = t, \neg b(e) = t\big). \tag{12}$$

A Naive Bayes model learned from the data of Table 2 is shown in the top of Fig. 4, with a corresponding TET below. Again the TET nodes are labeled with weight assignments corresponding to empirical frequencies $n^+/(n^+ + n^-)$.

Now the example $a(e) = t$, $b(e) = f$, $c(e) = t$ evaluates to the TET value

$$V\big(T(e)\big) = \big(t, \{t\}, \{f\}, \{f\}, \{t\}, \{t\}, \{f\}\big),$$

which gives a discriminant function value equal to

$$
\begin{aligned}
d_+^{\boldsymbol{\beta}}\big(V\big(T(e)\big)\big) &= 50/100\,\frac{10/40}{50/100}\,\frac{30/46}{50/100}\,\frac{31/64}{50/100} \\
&= P(+)\,\frac{P(+\mid a(e)=t)}{P(+)}\,\frac{P(+\mid b(e)=t)}{P(+)}\,\frac{P(+\mid c(e)=t)}{P(+)} \\
&= P(+)\,\frac{P(a(e)=t\mid +)}{P(a(e)=t)}\,\frac{P(\neg b(e)=t\mid +)}{P(\neg b(e)=t)}\,\frac{P(c(e)=t\mid +)}{P(c(e)=t)}. \qquad (13)
\end{aligned}
$$

While (13) is not quite equivalent to the posterior probability $P(+\mid a(e)=t,\neg b(e)=t,c(e)=t)$ that one would obtain from the Naive Bayes model, one still obtains that the ratio $d_+^{\boldsymbol{\beta}}(V(T(e)))/d_-^{\boldsymbol{\beta}}(V(T(e)))$ (with $d_-^{\boldsymbol{\beta}}$ similarly defined as $d_+^{\boldsymbol{\beta}}$, only using the negative class frequencies $n^-/(n^+{+}n^-)$ as weights) is equal to the odds ratio $P(+\mid a(e),b(e),c(e))/P(-\mid a(e),b(e),c(e))$ in the Naive Bayes model.

The special cases (12) and (13) for TETs emulating decision trees and Naive Bayes models, respectively, generalize to arbitrary propositional TETs as follows. To simplify matters, consider a TET T with a single free variable X, and vacuous root $\top(X)$. Assume that T has n nodes besides the root, and that the ith node is labeled with a type $\tau_i(X)$. We denote with $\phi_i^p(X)$ the Boolean feature that is the conjunction of all types on the path from the root to node i (not including $\tau_i(X)$ itself). Let node i be labeled with the weight $P(+\mid \phi_i^p=t,\tau_i=t)$.

Now consider an example e. It defines a "prefix" of T consisting of all nodes i for which $\phi_i^p(e)\wedge\tau_i(e)$ evaluates to t. Without loss of generality, assume that the nodes in this prefix are just $i=1,\ldots,l$ for some $l\leqslant n$. Then the discriminant function value for e is

$$
d_+^{\boldsymbol{\beta}}\big(V\big(T(e)\big)\big) = P(+)\prod_{i=1}^{l}\frac{P(+\mid \phi_i^p=t,\tau_i=t)}{P(+\mid \phi_i^p=t)} = P(+)\prod_{i=1}^{l}\frac{P(\tau_i=t\mid +,\phi_i^p=t)}{P(\tau_i=t\mid \phi_i^p=t)}.
$$

Let $k_i\geqslant 0$ be the number of children of node $i\leqslant l$ within the prefix $\{1,\ldots,l\}$ (k_i will usually be less than the actual number of children of τ_i in the full TET T). For $j=1,\ldots,k_i$ let ϕ_i^j be the Boolean feature that is the conjunction of all prefix nodes contained in the subtree rooted at the jth child of i (note that ϕ_i^j depends on e, which defines the relevant prefix). If we now assume that for each $i\leqslant l$ the child features ϕ_i^j ($j=1\ldots k_i$) are independent given $\phi_i^p\wedge\tau_i$ and $class=+$, then one obtains

$$
P(+)\prod_{i=1}^{l}P\big(\tau_i=t\;\big|\;+,\phi_i^p=t\big) = P\Bigg(+,\bigwedge_{i=1}^{l}\tau_i=t\Bigg),
$$

and from that

$$
\frac{d_+^{\boldsymbol{\beta}}(V(T(e)))}{d_-^{\boldsymbol{\beta}}(V(T(e)))} = \frac{P(+,\bigwedge_{i=1}^{l}\tau_i=t)}{P(-,\bigwedge_{i=1}^{l}\tau_i=t)}. \qquad (14)
$$

Thus, in this way $d_+^{\boldsymbol{\beta}}/d_-^{\boldsymbol{\beta}}$ can be interpreted as an odds ratio. The independence assumption we made to arrive at this interpretation is appropriate for the decision tree and Naive Bayes emulating TETs: in the first case, it actually becomes vacuous, because here the prefix defined by an example always consists of a single branch. For the Naive Bayes TETs, it is just the regular Naive Bayes assumption. For TETs that do not have a pure decision tree or Naive Bayes structure, the independence assumption can still be reasonable, and lead to a coherent probabilistic interpretation of the d_+/d_- ratio. Note, however, that one also easily can construct TETs in which our independence assumption is infeasible, due to logical dependencies between different child features ϕ_i^j. Thus, while our analysis here leads to a general understanding of the nature of the discriminant function, it does not necessarily endow it in all cases with a coherent probabilistic semantics.

So far, we have considered propositional TETs only. However, the analysis of the discriminant function for this case directly carries over to non-propositional TETs. For this we only have to observe that for any given concrete domain M one can transform a TET T into an equivalent propositional one by grounding all variable-introducing edges, i.e., by replacing a branch $\xrightarrow{\boldsymbol{W}} T'(\boldsymbol{V},\boldsymbol{W})$ with the set of branches $\rightarrow T'(\boldsymbol{V},\boldsymbol{a})$ ($\boldsymbol{a}\in M^{|\boldsymbol{W}|}$). If all the groundings of the original branch are labeled with copies of the original weight assignment, then the discriminant function defined by the grounded TET is the same as the discriminant function defined by the original TET. Thus, the interpretation of the discriminant function on propositional TETs also explains the discriminant function on general TETs, with two additional assumptions, or observations: our general independence assumption implies that features $T'(\boldsymbol{V},\boldsymbol{a}),T'(\boldsymbol{V},\boldsymbol{a}')$ defined by two different substitutions of constants in a sub-TET $T'(\boldsymbol{V},\boldsymbol{W})$ are assumed as independent, and the d_+/d_- ratio one obtains is only an approximation of (14), since the weights defined by the initial TET are not exact class frequencies for the ground features, but only shared approximations obtained from aggregating statistics from all groundings (see Section 5).

Example 3.2. Consider the following weight assignment for the TET from Example 2.3:

$$
\mathit{author}(A) \overset{1.0}{\xrightarrow{P_1}} \mathit{authorOf}(A,P_1) \overset{1.5}{\xrightarrow{P_2}} \mathit{cites}(P_2,P_1).
$$

We compute the discriminant function value for γ_2 and γ_3 from Example 2.6:

$$\gamma_2 : \left(t, \left\{ \begin{array}{l} f:7 \\ (t, \{f:9, t:1\}):2 \\ (t, \{f:10\}):1 \end{array} \right\}\right) \qquad \gamma_3 : \left(t, \left\{ \begin{array}{l} f:7 \\ (t, \{f:8, t:2\}):1 \\ (t, \{f:10\}):2 \end{array} \right\}\right).$$

Since the underlying TET consists of a single branch, here $m = 1$ at all levels in the recursive value definition. At the first level of recursion we obtain:

$$d^{\boldsymbol{\beta}}(\gamma_2) = 1.0 \cdot \left(d^{\boldsymbol{\beta}}\left(\left(t, \{f:9, t:1\}\right)\right)\right)^2 \cdot d^{\boldsymbol{\beta}}\left(\left(t, \{f:10\}\right)\right). \tag{15}$$

To proceed, we compute for values of the form $(t, \{f:k, t:l\})$ for the sub-TET $T'(A, P_1)$ (cf. (5)):

$$d^{\boldsymbol{\beta}}\left(\left(t, \{f:k, t:l\}\right)\right) = 1.5 \cdot \left(\frac{2.0}{1.5}\right)^l.$$

Plugging this into (15) gives

$$d^{\boldsymbol{\beta}}(\gamma_2) = 1.0 \cdot \left(1.5 \cdot \left(\frac{2.0}{1.5}\right)^1\right)^2 \cdot 1.5 = (1.5)^3 \cdot \left(\frac{2.0}{1.5}\right)^2. \tag{16}$$

The last re-arrangement of the terms in (16) can be read as follows: each paper P_1 written by author A contributes a factor 1.5 to the discriminant function value, and each citation to a paper by A contributes a factor of 2.0/1.5. Since in γ_3 the total number of authored papers, and citations to these papers also is 3, respectively 2, the same discriminant function value is obtained:

$$d^{\boldsymbol{\beta}}(\gamma_3) = 1.0 \cdot 1.5 \cdot \left(\frac{2.0}{1.5}\right)^2 \cdot 1.5^2 = (1.5)^3 \cdot \left(\frac{2.0}{1.5}\right)^2. \tag{17}$$

The preceding example points to a limitation of the discriminant function: the function value $d(\gamma)$ depends only on certain "flat" counts contained in γ, not on the more detailed count-of-count structure. On the other hand, being a product of factors determined by simple counts, it turns out that TETs with the discriminant function can emulate Markov Logic Networks, as the following example illustrates.

Example 3.3. Consider a MLN knowledge base (9) in which all ϕ_i are conjunctions of literals. The MLN then defines the weight of a model $\mathcal{M}$ as

$$e^{\sum_{i=1}^n count\{\boldsymbol{X}_i | \phi_i(\boldsymbol{X}_i)\}[\mathcal{M}] w_i} = \prod_{i=1}^n \left(e^{w_i}\right)^{count\{\boldsymbol{X}_i | \phi_i(\boldsymbol{X}_i)\}[\mathcal{M}]},$$

where $count\{\boldsymbol{X}_i \mid \phi_i(\boldsymbol{X}_i)\}[\mathcal{M}]$ is the value of the count feature (10) in $\mathcal{M}$.

The same weight function on models is defined by the discriminant function on the TET

$$\left[\top(), \left(\boldsymbol{X}_1, \phi_1(\boldsymbol{X}_1)\right), \ldots, \left(\boldsymbol{X}_n, \phi_1(\boldsymbol{X}_n)\right)\right]$$

with the weight assignment $\boldsymbol{\beta} = (1, e^{w_1}, \ldots, e^{w_n})$.

MLNs whose formulas ϕ_i are arbitrary quantifier-free formulas also can be emulated by a TET discriminant function. For this one may write ϕ_i in a disjunctive normal form $\bigvee_j \bigwedge_k l_{ijk}$ with literals l_{ijk}, such that the individual disjuncts $\phi_{ij} := \bigwedge_k l_{ijk}$ are mutually exclusive. Then the same construction as above applied to all formulas ϕ_{ij} with associated weights w_i yields a discriminant function representation of the MLN weight function. This representation, however, may now be of a size that is exponential in the length of the original formulas ϕ_i.

4. TET metric

After the simple discriminant function of the previous section, we now introduce our second tool to build predictive and descriptive models directly on TET-defined features. This consists of the definition of a metric on the value space $\mathcal{V}(T)$ of a TET T.

The metric is defined by induction on the structure of $\mathcal{V}(T)$. Following Definition 2.5, the base case is:

- If $\mathcal{V}(T) = \{t, f\}$, define $d_{tet}(t, f) = 1$, $d_{tet}(t, t) = d_{tet}(f, f) = 0$.

The core of the induction step for d_{tet} consists of the specification of the distance between two multisets of values

$$\mu = (\gamma_1 : k_1, \ldots, \gamma_m : k_m),\ \mu' = \left(\gamma'_1 : k'_1, \ldots, \gamma'_{l'} : k'_l\right)$$

where the γ_i, γ'_j all come from a value space $\mathcal{V}(\tilde{T})$ of some sub-TET $\tilde{T}$ of T, and d_{tet} is already defined on $\mathcal{V}(\tilde{T})$.

After normalizing the counts to probability values $p_i := k_i/(k_1 + \cdots + k_m)$, $p'_j := k'_j/(k'_1 + \cdots + k'_l)$ we can view the two values as probability distributions on the metric space $(\mathcal{V}(\tilde{T}), d_{tet})$. A standard way to define a metric on such distributions is the well-known *Wasserstein–Kantorovich* or *Earth-Mover's Distance*:

Definition 4.1. Let $\mu = (\gamma_1 : p_1, \ldots, \gamma_m : p_m)$, $\mu' = (\gamma'_1 : p'_1, \ldots, \gamma'_{l'} : p'_l)$ be two multisets over $\mathcal{V}(\tilde{T})$ with counts normalized to probability distributions p, p'.

Let d_{tet} be a metric on $\mathcal{V}(\tilde{T})$. The Wasserstein–Kantorovich distance between μ and μ' is

$$d_{WK}(\mu, \mu') = \inf_q \sum_{i,j} d_{tet}(\gamma_i, \gamma'_j) q(\gamma_i, \gamma'_j)$$

where the infimum is taken over all probability distributions q on $\mathcal{V}(\tilde{T}) \times \mathcal{V}(\tilde{T})$ whose marginal on the first component is equal to p, and whose marginal on the second component is equal to p'. Note that d_{WK} is a valid metric if the "ground distance" d_{tet} is a valid metric [4].

We now define $d_{tet}(\gamma, \gamma')$ for $\gamma, \gamma' \in \mathcal{V}(T)$:

Definition 4.2. Let $\mathcal{V}(T) = \{f\} \cup \{t\} \times \bigtimes_{i=1}^m multisets(\mathcal{V}(T_i))$, and assume that d_{tet} is defined on $\mathcal{V}(T_i)$ $(i = 1, \ldots, m)$. Let $\gamma, \gamma' \in \mathcal{V}(T)$. Depending on whether one or both of γ, γ' are f, we define $d_{tet}(\gamma, \gamma')$ as:

γ' \ γ	f	$\neq f$
f	0	1
$\neq f$	1	Eq. (19)

(18)

In the non-trivial case $\gamma, \gamma' \neq f$, we have $\gamma = (t, \mu_1, \ldots, \mu_m)$, $\gamma' = (t, \mu'_1, \ldots, \mu'_m)$ with

$$(\mu_1, \ldots, \mu_m), \left(\mu'_1, \ldots, \mu'_m\right) \in \bigtimes_{i=1}^m multisets\left(\mathcal{V}(T_i)\right).$$

Then

$$d_{tet}(\gamma, \gamma') := \sum_{i=1}^m \omega_i d_{WK}(\mu_i, \mu'_i), \tag{19}$$

where $\omega_0, \ldots, \omega_m > 0$ are adjustable weight parameters with $\sum_i \omega_i = 1$.

Proposition 4.3. *For all T, d_{tet} is a metric on $\mathcal{V}(T)$ with values in* $[0, 1]$.

Proof. The statement is clearly true for the base case $\mathcal{V}(T) = \{t, f\}$.

For the case $\mathcal{V}(T) = \{f\} \cup \{t\} \times \bigtimes_{i=1}^m multisets(\mathcal{V}(T_i))$ we have the induction hypothesis that d_{tet} is a metric with values in $[0, 1]$ on $\mathcal{V}(T_i)$ $(i = 1, \ldots, m)$. Then d_{WK} defined on multisets (μ_i, μ'_i) over $\mathcal{V}(T_i)$ is a metric for all $i = 1, \ldots, m$, and hence the convex combination in (19) defines a metric on $\mathcal{V}(T) \setminus \{f\}$. By the condition $\sum_i \omega_i = 1$ its values lie in the interval $[0, 1]$.

It remains to show that the extension via (18) to include the f case still satisfies the properties of a metric. $d_{tet}(\gamma, \gamma') \geqslant 0$ with equality only for $\gamma = \gamma'$, and $d_{tet}(\gamma, \gamma') \leqslant 1$, as well as symmetry $d_{tet}(\gamma, \gamma') = d_{tet}(\gamma', \gamma)$ are clearly satisfied. For the triangle inequality, consider $d_{tet}(\gamma, \gamma') + d_{tet}(\gamma', \gamma'')$. If $\gamma = \gamma' = \gamma'' = f$, then the sum is zero, and equal to $d_{tet}(\gamma, \gamma'')$. If one of $\gamma, \gamma', \gamma''$ is not f, then the sum is greater or equal 1, and thus greater or equal $d_{tet}(\gamma, \gamma'')$. □

The TET metric defined above can be computed using the *transportation simplex* algorithm, a specialized linear programming algorithm for solving the transportation problem [37]. Ling and Okada [25] have introduced a faster algorithm for computing the Earth Mover's Distance between histogram. However that algorithm assumes fixed-size histograms and cannot deal with signatures of distributions, as required by the recursive definition of d_{tet}.

In our experiments, we found the CPU time required for computing the distances to be negligible compared to CPU time for computing the TET-values. A simple theoretical analysis justifies this finding. Assume we want to calculate the distance between two TET values having (for simplicity) the same shape, uniform branching factor m, and height h. Let $n = m^h$ the number of nodes in the TET value and assume the transportation simplex (which needs to be computed on each TET-value

Table 3
Distance matrices for propositional TETs.

	$a_1\ a_2$	$a_1 \to a_2$	$a_1 \leftarrow a_2$	$a_1 \leftrightarrow a_2$
$a'_1\ a'_2$	0	1	0	0
$a'_1 \to a'_2$	1	0	1	1
$a'_1 \leftarrow a'_2$	0	1	0	0
$a'_1 \leftrightarrow a'_2$	0	1	0	0

(a)

	$a_1\ a_2$	$a_1 \to a_2$	$a_1 \leftarrow a_2$	$a_1 \leftrightarrow a_2$
$a'_1\ a'_2$	0	1	0	1
$a'_1 \to a'_2$	1	0	1	1
$a'_1 \leftarrow a'_2$	0	1	0	1
$a'_1 \leftrightarrow a'_2$	1	1	1	0

(b)

	$a_1\ a_2$	$a_1 \to a_2$	$a_1 \leftarrow a_2$	$a_1 \leftrightarrow a_2$
$a'_1\ a'_2$	0	1/2	1/2	1
$a'_1 \to a'_2$	1/2	0	1	1/2
$a'_1 \leftarrow a'_2$	1/2	1	0	1/2
$a'_1 \leftrightarrow a'_2$	1	1/2	1/2	0

(c)

	$a_1\ a_2$	$a_1 \to a_2$	$a_1 \leftarrow a_2$	$a_1 \leftrightarrow a_2$
$a'_1\ a'_2$	0	1/2	1/2	1/2
$a'_1 \to a'_2$	1/2	0	1/2	1/2
$a'_1 \leftarrow a'_2$	1/2	1/2	0	1/2
$a'_1 \leftrightarrow a'_2$	1/2	1/2	1/2	0

(d)

Table 4
Author distance matrix: (a): no normalization (b): false counts normalized.

	a_1	a_2	a_3	a_4	a_5
a_1	0	0.11	0.11	0.01	0.13
a_2		0	0.02	0.11	0.02
a_3			0	0.1	0.02
a_4				0	0.12
a_5					0

(a)

	a_1	a_2	a_3	a_4	a_5
a_1	0	0.26	0.37	0.23	0.59
a_2		0	0.22	0.19	0.32
a_3			0	0.14	0.22
a_4				0	0.36
a_5					0

(b)

node) takes a polynomial time $O(m^k)$ for some k (in [37] k was empirically found to be between 3 and 4). We therefore have the recurrence for the running time $T(n)$ of the TET-distance calculation:

$$T(n) = mT\left(\frac{n}{m}\right) + O\left(m^k\right) = mT\left(\frac{n}{m}\right) + O\left(n^{\frac{k}{h}}\right).$$

By the master theorem, if $h > k$ then $T(n) = O(n)$ and if $h < k$ then $T(n) = O(n^{\frac{k}{h}})$.

We now illustrate some of the properties of the d_{tet} metric. Our first example uses simple propositional TETs to illustrate the flexibility of the d_{tet} metric that derives from varying TET structures.

Example 4.4. Consider the four TETs (a)–(d) from Example 2.8. Table 1 gave for the four possible configurations of entity pairs a_1, a_2 the associated TET values. Table 3 now shows the distance matrices obtained from evaluating d_{tet} on these values. For (c) and (d) uniform weights for the different branches have been used, i.e., $\omega_i = 1/2$ in (c), and $\omega_i = 1/4$ in (d). For better readability and ease of comparison, the rows and columns in these matrices are indexed by the a_1, a_2-substructures, even though the entries in the table are, of course, a function of their values.

We obtain the following characteristics of the distance function defined by the four TETs:

(a) This is 0/1-distance to the "reference structure" $a_1 \to a_2$: any pair a'_1, a'_2 that has a different structure has distance 1 to $a_1 \to a_2$, and distance 0 to any other pair that also does not have the reference structure.
(b) This metric identifies two structures (a_1, a_2), (a'_1, a'_2) (i.e., assigns zero distance between them), if neither contains the edge $\to$. Otherwise, structures have distance 0 iff they are equal, and distance 1 else.
(c) Here the TET metric becomes the (normalized) edit distance relative to the two primitive edit operations edge insertion and edge deletion.
(d) This is a scaled 0/1 distance: two structures (a_1, a_2), (a'_1, a'_2) have a constant distance > 0 iff they are different. Note that two distinct structures have distance 1/2 rather than 1, because their values agree on the two out of four TET branches that evaluate to f for both of them. The two other branches each return a distance of 1, which with the $\omega_i = 1/4$ weights gives a total distance 1/2.

Example 4.5. Table 4(a) gives the distances between the values γ_i shown in (6), i.e. the distances between the authors a_i in Fig. 1 defined by the TET (1).

The matrix shows that according to d_{tet} there are two clusters $\{a_1, a_4\}$ and $\{a_2, a_3, a_5\}$ of authors: the distances between authors within each of these groups is about one order of magnitude smaller than the distance between authors from different groups.

Comparing with Fig. 1 one finds that the clusters are defined by the number of papers written by an author: two for the first cluster, and three for the second. Given the difference in the number of authored papers, the citation distribution has a secondary influence on the distance value: thus, for example, $d(a_1, a_2) < d(a_1, a_5)$, because the citation pattern of the two papers of a_1 is more similar to the one of the three papers of a_2, than the three papers of a_5.

The preceding example highlights a potential problem with the definition of d_{tet}: differences in top-level counts appear to have a dominating influence on the distance values. While it often will be reasonable that primary counts have a larger impact than counts in the lower levels of the TET, it may be desirable to control the extent to which this is happening. In the following we first analyze in a more general manner the distances obtained between certain TET values, and then introduce a method for adjusting the metric so that its behavior can be adapted to fit more closely the needs in specific applications.

We consider the generic two-level count-of-count TET

$$\top(V) \xrightarrow{W} r(V, W) \xrightarrow{U} s(W, U).$$

Assume that the variables W, U are typed, such that W ranges over a sub-domain of size K, and U ranges over a sub-domain of size N. We now consider values of the form

$$\left(t, \left\{ \begin{array}{l} f : K - k \\ (t, \{f : N - n, t : n\}) : k \end{array} \right\}\right).$$

These values are symmetric in the sense that all W with $r(V, W)$ have the same number n of $s()$-successors U (cf. also the values for a_2, a_4, a_5 in (6)). Assuming K, N to be fixed, these values are fully characterized by the two parameters n, k, and one can derive a closed-form expression for pairs of values of this form:

$$d_{WK}\big((n, k), (n', k')\big) = \frac{\min(k, k')}{K} \frac{|n - n'|}{N} + \frac{|k - k'|}{K}. \tag{20}$$

This expression explains several potential problems that were already visible in the example of Table 4(a): first, since typically $n, n' \ll N$ and $k, k' \ll K$, the distances will tend to be very small numbers. Second, the distance is dominated by the difference $\frac{|k-k'|}{K}$ in counts at the first level of the TET. Furthermore, the distance is sensitive to the sizes of the domains over which the variables range: the behavior of (20) as a function of the actual counts n, k, n', k' depends on K, N, in particular on differences in order of magnitude ($K \ll N$ or $N \ll K$). Note that the third issue is akin to the situation in standard attribute-value data, where certain numeric features may dominate a distance measure due to the order of magnitude of their measuring scale. We can address all these (potential) problems by introducing a normalization operation on TET values.

4.1. Value normalization

In analogy to standard normalization procedures for numeric data, we introduce a normalization operation for TET values. A standard normalization procedure for numeric data would be a linear transformation $x \mapsto ax + b$, where the coefficients a, b are such that the empirical distribution in the transformed dataset has mean 0 and variance 1. Note that here the concrete coefficients a, b depend on the dataset (the original empirical mean and variance of x), but that the normalization procedure in general is defined by the two "hyper-parameters" 0 and 1. Instead of standardizing all numeric attributes to 0 mean and variance 1, one could also assign different such hyper-parameters to different attributes, and thereby adjust the impact different attributes have on the overall distance function.

For TET values, we will perform normalization by scaling f counts, i.e., replacing occurrences of $f : k$ by $f : ak$ for some a. The normalization is guided by data-independent hyper-parameters that can be adjusted to optimize the behavior of the TET metric d_{tet} for specific purposes. The concrete multiplicative factors a will then depend on the hyper-parameters, and the empirical distribution of TET values for which the normalization is performed. The hyper-parameters are defined by a normalization labeling in the sense of the following definition.

Definition 4.6. A *normalization labeling* for a TET

$$T(\boldsymbol{V}) = \big[\tau(\boldsymbol{V}), \big(\boldsymbol{W}_1, T_1(\boldsymbol{V}, \boldsymbol{W}_1)\big), \ldots, \big(\boldsymbol{W}_m, T_m(\boldsymbol{V}, \boldsymbol{W}_m)\big)\big]$$

is given by a vector $(y_1, \ldots, y_m)$ of non-negative real numbers, and a normalization labeling for each of the sub-TETs T_i $(i = 1, \ldots, m)$.

In the following, we assume for notational convenience $m = 1$, i.e., $T = [\tau(\boldsymbol{V}), (\boldsymbol{W}, T'(\boldsymbol{V}, \boldsymbol{W}))]$.

Definition 4.7. Let $T = [\tau(\boldsymbol{V}), (\boldsymbol{W}, T'(\boldsymbol{V}, \boldsymbol{W}))]$, and $(y, \boldsymbol{y}')$ be a normalization labeling ($\boldsymbol{y}'$ a normalization labeling for T'). Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\} \subseteq \mathcal{V}(T) \setminus \{f\}$.

We write the γ_i as

$$\gamma_i = \left(t, \left\{ f : k_i^f, \gamma_{i,1} : k_{i,1}, \ldots, \gamma_{i,l_i} : k_{i,l_i} \right\}\right)$$

with $\gamma_{i,j} \neq f$.

Define

$$k_{avg}^f = 1/n \sum_{i=1}^{n} k_i^f, \qquad k_{avg}^{\neq f} = 1/n \sum_{i=1}^{n} \sum_{j=1}^{l_i} k_{i,j}.$$

The *normalization* of γ_i in Γ with hyper-parameters $(y, \boldsymbol{y}')$ is now given by:

- replacing k_i^f by $y(k_{avg}^{\neq f}/k_{avg}^f)k_i^f$.
- replacing each $\gamma_{i,j}$ by the normalization of $\gamma_{i,j}$ in $\Gamma' := \{\gamma_{i,j} \mid i = 1, \ldots, n;\ j = 1, \ldots, l_i\}$ with hyper-parameters $\boldsymbol{y}'$.

A normalization parameter y specifies the ratio of total f to non-f counts in all the values in the given dataset corresponding to the branch labeled with y.

Example 4.8. A normalization labeling for our basic bibliographic TET is

$$author(A) \xrightarrow{P_1, \mathbf{0.1}} authorOf(A, P_1) \xrightarrow{P_2, \mathbf{1.0}} cites(P_2, P_1).$$

The hyper-parameters $(0.1, 1.0)$ were found in our experiments on predicting the h-index in the DBLP dataset (cf. Section 6.1). Normalizing the dataset consisting of the 5 values in (6) gives the normalized values:

$$\gamma_1^*\colon \left(t, \left\{ \begin{array}{l} f : 0.281 \\ (t, \{f : 0.878, t : 1\}) : 1 \\ (t, \{f : 0.78, t : 2\}) : 1 \end{array} \right\}\right)$$

$$\gamma_2^*\colon \left(t, \left\{ \begin{array}{l} f : 0.246 \\ (t, \{f : 0.878, t : 1\}) : 2 \\ (t, \{f : 0.976\}) : 1 \end{array} \right\}\right)$$

$$\gamma_3^*\colon \left(t, \left\{ \begin{array}{l} f : 0.246 \\ (t, \{f : 0.78, t : 2\}) : 1 \\ (t, \{f : 0.976\}) : 2 \end{array} \right\}\right)$$

$$\gamma_4^*\colon \left(t, \left\{ \begin{array}{l} f : 0.281 \\ (t, \{f : 0.78, t : 2\}) : 1 \\ (t, \{f : 0.976\}) : 1 \end{array} \right\}\right)$$

$$\gamma_5^*\colon \left(t, \left\{ \begin{array}{l} f : 0.246 \\ (t, \{f : 0.976\}) : 3 \end{array} \right\}\right).$$

Observe that $0.1 = (0.281 + 0.246 + 0.246 + 0.281 + 0.246)/(2 + 3 + 3 + 2 + 3)$, and $1 = (2 \cdot 0.878 + 3 \cdot 0.78 + 4 \cdot 0.976)/(1 + 2 + 1 + 2 + 2)$.

The distance matrix obtained for these normalized values is shown in Table 4(b). One immediately sees that now the range of distance values is more spread out in the interval $[0, 1]$, and that the clustering according to paper count has disappeared. The most dis-similar authors now (as before) are a_1 and a_5 (few papers with many citations vs. many papers without citations). However, the most similar authors now are a_3 and a_4, who previously even belonged to different clusters. Seeing that a_3 differs from a_4 only by the addition of one paper without citations, it makes intuitive sense that the distance measure optimized for predicting the h-index sees them as nearest neighbors.

5. TET learning

We first describe in more detail the learning problem we want to solve. Our data consists of a model $\mathcal{M}$ in the sense of Definition 2.1. In our implementation, $\mathcal{M}$ is given as a relational database containing one table for each $r \in R$, where the table for r contains all tuples $\boldsymbol{a} \in M^{arity(r)}$ for which $I(r(\boldsymbol{a})) = true$. Furthermore, we are given an initial *target table*, i.e. a table consisting of a set of examples with $+/-$ class labels. For example, a learning problem given by the data depicted in Fig. 1, with a_1, a_3, a_4 as positive and a_2, a_5 as negative examples would be given by the 4 leftmost tables in Table 5. Columns in the data tables are headed by synthetic identifiers Arg_i. Columns in the target table (other than the class label

Table 5
Input data tables.

Arg_1
a_1
a_2
⋮
a_5
author

Arg_1
p_1
p_2
⋮
p_{10}
paper

Arg_1	Arg_2
a_1	p_1
a_1	p_2
⋮	⋮
a_5	p_{10}
authorOf	

Arg_1	Arg_2
p_3	p_1
p_5	p_2
⋮	⋮
p_9	p_6
cites	

A	Label
a_1	+
a_2	−
a_3	+
a_4	+
a_5	−
target	

Table 6
Local target table.

A	P	Label
a_1	p_1	+
a_1	p_2	+
a_2	p_3	−
⋮	⋮	⋮
a_5	p_8	−
a_5	p_9	−
a_5	p_{10}	−
target		

column) are headed by variable names, which will then become the names of the free variables in the TET we construct. Thus, given the input we will want to construct a TET $T(A)$ over the signature $R = \{author, paper, authorOf, cites\}$ that captures features of A that are predictive for the class label given in *target*.

Our general approach to TET learning is a recursive top-down construction that associates with each node a local discrimination task represented by a local target table. In our running example, starting with the input data in Table 5 we would initialize the TET construction with the vacuous TET $\top(A)$. If the first extension was $\top(A) \xrightarrow{P} authorOf(A, P)$, then we would associate with the node $authorOf(A, P)$ the local target table shown in Table 6. This construction of local target tables is essentially the same as the construction of local training sets in FOIL [34]. The construction of this new target table amounts to a problem transformation: the problem of predicting the label of an author is transformed into predicting the label of author/paper pairs in the new target table, which may be effected by taking into consideration attributes of both authors and papers, as well as additional relations between authors and papers (if any such exist in the data).

The exact specification of the construction of local target tables is as follows. Let n be a TET node associated with a local target table $tt_n(\boldsymbol{V}, L)$ with columns for variables $\boldsymbol{V}$ and label L. Let n' be a child of n labeled with type $\sigma(\boldsymbol{V}, \boldsymbol{W})$, and reached by an edge labeled with variables $\boldsymbol{W}$ (we include the possibility that $|\boldsymbol{W}| = 0$, i.e. the edge is really unlabeled). Then n' is associated with a target table $tt_{n'}$ with columns for $\boldsymbol{V}, \boldsymbol{W}$ and L, defined as:

$$tt_{n'}(\boldsymbol{V}, \boldsymbol{W}, L) = \left\{(\boldsymbol{a}, \boldsymbol{b}, l) \in M^{|\boldsymbol{V}|+|\boldsymbol{W}|} \times \{+, -\} : (\boldsymbol{a}, l) \in tt_n;\ I\big(\sigma(\boldsymbol{a}, \boldsymbol{b})\big) = true\right\}. \tag{21}$$

In the case $|\boldsymbol{W}| = 0$, $tt_{n'}$ is just the subset of tt_n containing the elements for which $\sigma(\boldsymbol{a})$ is true.

When building the TET, candidate tree-extensions $\ldots \xrightarrow{\boldsymbol{W}} \sigma(\boldsymbol{V}, \boldsymbol{W})$ are scored based on the *relational information gain* (*RIG*) measure proposed in [26]. *RIG* values represent both *direct* and *potential* informativeness of the extension: direct informativeness is provided by extensions that in one step increase the class-purity of the local target table. Potential informativeness is provided by extensions that introduce new entities into the target table which in subsequent steps might enable discrimination between positive and negative examples via additional features related to the new entities.

High *RIG* values (unlike information gain in decision tree learning, for example), thus, do not give bounds on a guaranteed improvement of classification accuracy in a single construction step, but may only indicate a potential improvement that could be obtained by further construction steps. After termination of a recursive sub-tree construction, therefore, the final predictive accuracy gain of the sub-tree is evaluated, and the sub-tree is pruned if this gain does not exceed a given threshold. The evaluation of the current TET's accuracy has to be based on a concrete classification model built on the TET feature. It is here that we crucially use the discriminant function model of Section 3: a weight assignment defining the discriminant function is very fast to learn, and the resulting discriminant function values on the validation set are fast to compute. The overall wrapper-evaluation with the discriminant function, thus, is computationally very efficient. If the final TET is to be used in conjunction with a different classification model than the discriminant function, then it could be beneficial to already use that classification model in the wrapper-evaluation during TET learning. However, for complex models this can be computationally very expensive. Furthermore, it does not appear to be the case that TET features learned using the discriminant function are highly biased towards this particular classification model. The discriminant function, thus, can play the role of a lightweight classification model that may be used to discover features which are also useful for more complex model types, similarly as in the propositional case decision tree or logistic regression model learning can act as a feature selector also for subsequent use in more complex models, like support vector machines (cf. Fig. 2).

Table 7
TET learning.

build_TET(*Data* $\mathcal{M}$, *Labeled table* tt, *TET_node parent*, *TET_node root*)
1. *parent*.weight = *positive_class_frequency*(tt)
2. *current_score* = *predictive_score*($\mathcal{M}$, *root*)
3. *EXT*:=*possible_extensions*(*parent*, θ_{vars}, θ_{depth})
4. **for** all $\sigma(\boldsymbol{V}, \boldsymbol{W}) \in EXT$ compute $RIG(tt, \sigma(\boldsymbol{V}, \boldsymbol{W}))$
5. *CAND*:= *candidate_extensions*(*EXT*,*RIG*-values, θ_{RIG})
6. **for** all $\sigma(\boldsymbol{V}, \boldsymbol{W}) \in CAND$
7. $tt' = construct\_tt(\mathcal{M}, tt, \sigma(\boldsymbol{V}, \boldsymbol{W}))$
8. *nextChild* = new TET_node($\sigma(\boldsymbol{V}, \boldsymbol{W})$)
9. add *nextChild* as child to *parent*
10. *build_TET*($\mathcal{M}, tt'$, *nextChild*, *root*)
11. *new_score* == *predictive_score*($\mathcal{M}$, *root*)
12. **if** *new_score* − *current_score* $< \theta_{score}$
13. remove *nextChild* from *parent*
14. **else** *current_score*=*new_score*

Table 7 outlines the TET learning algorithm. It is implemented as a procedure that recursively expands an initial TET. It receives as arguments the data, a local target table tt to be classified, a pointer *parent* to the current node to be expanded and a pointer to the root of the TET being constructed. The initial call is

$$build\_TET(\mathcal{M}, tt, T, T)$$

where $T = new\ TET\_Node(\top(\boldsymbol{V}))$ is a pointer to an initial TET with vacuous root type $\top(\boldsymbol{V})$. The construction works as follows: Line 1 sets the weight for the discriminant function d_+ for the input node. It is just the relative frequency of positive examples in the target table associated with this node (the weight for d_- being one minus this weight). Thus, the discriminant function here is learned (at little extra cost) in parallel with the TET construction.

The function *predictive_score*($\mathcal{M}$, *root*) called in lines 2 and 11 performs the global evaluation of the current TET based on its predictive performance in conjunction with the chosen classification model. If a model other than the discriminant function here is used, then calls to *predictive_score*($\mathcal{M}$, *root*) may require computationally expensive model training for the current TET.

Lines 3–5 are crucial: here a subset of all the possible extensions of the current node defined by types $\sigma(\boldsymbol{V}, \boldsymbol{W})$ of child nodes is constructed for further exploration. This operation is analogous to refinement operators in ILP. Our construction is in two steps: in the first step the set of possible extensions for the current node is constructed by the function *possible_extensions*. This function can implement various constraints and a language bias. In our implementation and experiments, we restrict possible extensions in terms of the number of literals and the number of new variables in $\sigma(\boldsymbol{V}, \boldsymbol{W})$ (mostly limiting both numbers to at most one). The function can also take TILDE-style user-defined *rmode* declarations [3], that can force certain arguments of the new literal to be filled with variables already present in the parent node (input variable), or with a new variable introduced by this extension (output variable). As common in ILP algorithms, *type* predicates can be used to specify the type of arguments of the literals. In this case the learner expects a unary predicate for each variable type, being true for all and only the objects of that type. In addition to this user-defined bias, we force candidate extensions to use at least one of the latest introduced variables along their path to the root. The rationale for this constraint is that extensions introducing new variables were selected based on their RIG score and thus likely on their potential rather than direct informativeness. By focusing the search toward further refinements of the new variables, we force the algorithm to try making this potential informativeness explicit. When introducing a new variable, the algorithm automatically adds inequality constraints guaranteeing that they cannot be bound to the same value as that of any of the root variables, i.e. those identifying the entity the TET will represent features of. In case of a typed language bias, inequality constraints are added for variables of the same type only. In order to control the computational cost of adding new variables, we constraint the number of variables in each path from the root to a leaf to be within a user-defined maximum value (θ_{vars}). Finally, *possible_extensions* is used to implement a termination condition: if the depth of the current *parent* node in the TET has reached a (user specified) maximum depth (θ_{depth}), then *possible_extensions* will return an empty set. In the next step, the relational information gain is computed for all possible extensions; the function *candidate_extensions* then performs a selection based on *RIG* values. Our current implementation of *candidate_extensions* selects all extensions whose *RIG* value exceeds a user defined threshold (θ_{RIG}).

A child node is then created for each candidate extension. The function *construct_tt* constructs the local target table for the child according to (21). Lines 8–9 add a new child labeled with the current candidate extension $\sigma(\boldsymbol{V}, \boldsymbol{W})$ to *parent*. Line 10 continues the recursive construction at the new child, which then becomes the root of a whole new subtree. Lines 11–14 then evaluate the extension of the old TET with this new subtree, and either accept or reject it based on a user defined threshold for the required global score improvement (θ_{score}).

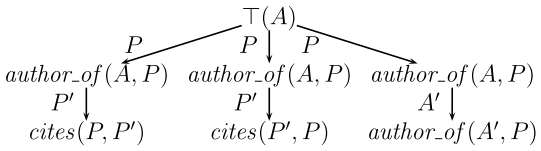


Fig. 5. The TET learned on the DBLP data set for the task of h-index prediction. The first branch on the left represents the relational features which are necessary and sufficient to compute h-indices.

6. Experiments

6.1. DBLP

As already stated in the introduction, bibliometrics represents an ideal domain to test the capability of TETs of learning count-of-counts features: in particular, we focus on the task of predicting h-indices. The h-index of an author A is defined as the maximum number h such that A authored h papers having at least h citations each [14].

The data set used in our experiments is taken from the DBLP Computer Science Bibliography [24][2] enhanced with citation data [41]. We extracted a set of facts and relations in the form of a MySQL database from the original data set available at http://www.arnetminer.org/citation. For the sake of reproducibility, we also provide a package with the scripts used to build the data set used in our experiments (see Supplementary Materials). From the original database we extracted the following tables:

- *author*(A) providing the id of each author;
- *paper*(P) providing the id of each paper;
- *author_of*(A, P), true if A is an author of P;
- *cites*(P_1, P_2), true if P_1 cites P_2.

Our first goal was to learn a TET able to discriminate between authors with high and low h-index (i.e., above/below a certain threshold). To this end, we extracted a sub-graph of the whole DBLP network (see Supplementary Materials) consisting of 8726 authors and 244,265 papers, and used the learning algorithm described in Table 7, employing relational information gain [26] as the scoring function guiding the search, and the discriminant function of Definition 3.1 to evaluate the TET score after the introduction of each new literal.[3] We chose an h-index threshold $h = 7$ to define positive and negative examples. We fixed $\theta_{RIG} = 0$, so that each candidate extension with non-zero RIG was considered (in the order given by the RIG score), $\theta_{score} = 1e - 4$ in order to prune away branches with low improvement, $\theta_{depth} = 2$ and $\theta_{vars} = 3$. Fig. 5 shows the learned TET: it consists of three different branches, the first corresponding to the count-of-counts feature which can be used to exactly compute the h-index, and the two other branches describing features which are also correlated to the h-index of author A: the number of papers P' cited by each paper P written by A (second branch), and the number of co-authors A' for each paper P of A (third). One may wonder why the learner did not return a TET consisting only of the first branch, since this feature would be sufficient to predict the h-index in the training examples exactly. Note however that the classification accuracy on the training examples is also constrained by the prediction model that maps the TET feature value into a binary classification. The discriminant function used by the learner (as well as virtually any other conceivable prediction model) is not able to exactly map the feature value obtained from the first branch into the binary threshold function $h > 7$, and therefore additional features represented by the additional branches can still be useful for obtaining a better fit to this threshold.

The TET shown in Fig. 5 in combination with the discriminant function guiding the learning phase achieves an F1 of 61.3% for the binary classification task of predicting authors with $h > 7$. As a comparison, we employed TILDE [3] to inductively learn a logical decision tree in the same setting. The task turned out to be quite difficult for TILDE: being the *author_of*(A, P) predicate only *potentially informative* but not *directly informative* [26], with the same language bias used by our TET learner, TILDE ended up the search with an empty tree. We therefore tried to modify the language bias, by allowing the joint introduction of the predicates *author_of*(A, P) and *cites*(P_1, P_2): the result was a quite complex tree learned by TILDE, which anyhow contained only three positive leaves covering a few examples, heading to an F1 of 1.8 %. We also tried to define some aggregates[4] in the language bias, counting the number of papers of an author and the number of citations of a paper. We used the TILDE option which allows the introduction of multiple aggregates within the same tree branch,[5] but in this case the search could not be completed due to memory requirements. We finally tried to use exhaustive lookahead, which turned out to be computationally very expensive: with only one level of lookahead, TILDE ran for over 20 days without terminating. As a comparison, the TET learning algorithm ran for about 7 minutes. Finally, in order to assess the

[2] http://dblp.uni-trier.de.

[3] No particular rmode declaration was specified.

[4] Aggregates are specific rmodes for TILDE defining predicates that represent aggregating functions, such as count, average, min, max.

[5] This option is called `aggregate_refinement`.

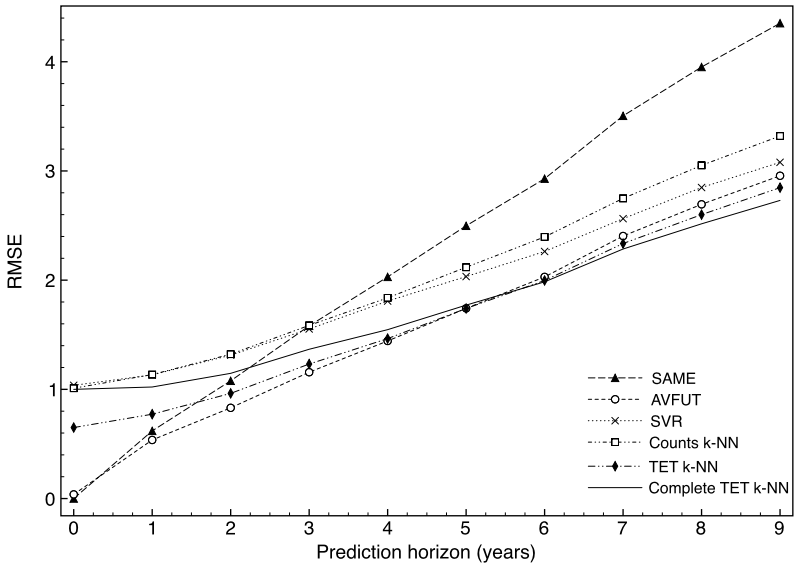


Fig. 6. Results on the task of h-index prediction over years: we plot the RMSE of several competitors as a function of the prediction year.

potential of the TET metric introduced in Section 4, we also tested our TET with a k-NN classifier in a leave-one-out setting, using the TET metric between TET values: in this case, the F1 achieved by the classifier, even without value normalization, was 88.5% with a single neighbor, and up to 91.2 with $k = 5$.

The second experiment is about h-index forecasting: given data up to a given year Y_0, we predict the h-index of an author in the forthcoming years. In this case, our aim was to measure the discriminative power of the metric defined on TET values and described in Section 4, with respect to several baselines and other prediction models using plain counts. The experimental framework is constructed as follows: first, we extracted from the whole DBLP the set of 8441 authors having h-index > 3 in $Y_0 = 2000$, and then split this set into 2/3 for the development set, and 1/3 for the test set (the development set was again split in 2/3 for training and 1/3 for validation). Our predictor was built as follows: using just the simple TET shown in the first branch of Fig. 5, which describes all the sufficient features to calculate h-index, we computed TET values for all the authors, and then run a simple k-NN algorithm employing the TET metric as distance between such values. This predictor was compared against several different competitors:

1. predict future h-index as equal to current h-index (SAME);
2. predict future h-index of a test author a as the average of future h-indices of training authors having current h-index equal to a (AVFUT);
3. predict future h-index using a k-NN algorithm, using as distance a linear combination of plain counts features, that is the number of papers n_{pap} and the number of citations n_{cit} ($h_{pred} = w_{pap} \times n_{pap} + w_{cit} \times n_{cit}$) ($k$-NN counts);
4. predict future h-index as a non-linear Support Vector Regressor taking plain counts features n_{pap} and n_{cit} as input (SVR);
5. predict future h-index using a k-NN algorithm with the TET metric, but using the whole learned TET (represented in Fig. 5) to compute TET values (complete TET k-NN).

Note that albeit very simple, the SAME and AVFUT predictors have a significant advantage over the other methods, since they make direct use of the h-index as a materialized feature in the data, whereas the other methods only are given underlying paper and citation counts.

The validation set was used to perform model selection over the parameters of each model: regularization parameter C and Gaussian kernel width γ were tuned for SVR (C ranges in $10^{-2}, \ldots, 10^2$, while γ ranges in $10^{-4}, \ldots, 1$); w_{pap}, w_{cit} (both ranging in $10^{-2}, \ldots, 10^2$) and the number of neighbors k for the counts-based k-NN; the normalization coefficients $\boldsymbol{y}$ and the number of neighbors k for k-NN with TET metric.[6] Root Mean Squared Error (RMSE) was used to measure the performance of each predictor.

Fig. 6 shows the results obtained as a function of the prediction horizon H over years, starting from 2000 – the year for which TET values were computed, and therefore corresponding to $H = 0$ – up to 2009 ($H = 9$).

The SAME and AVFUT predictors, by construction, have 0 error at $H = 0$, and remain very accurate for short prediction horizons, as the dynamics of h-indices change slowly over time. The TET based predictor has a lower average prediction error starting from $H > 5$, and always outperforms methods based on plain counts of features. It should also be noticed that

[6] Model selection for the normalization coefficients was performed on the simple TET, and resulted in the normalization labeling shown in Example 4.8.

Table 8
F1 results on the CORA dataset. Horizontal lines separate different learning parameter settings (allowing for increasingly complex TETs to be learned), plus a manually curated TET T (hand). Rows within each learning setting indicate a different predictive model, either discriminant function (DF) or k-NN with the TET metric, for different values of $k > 1$ (only values producing non-negligible differences are reported). Columns report results for each fold and macro-averaged over the five folds.

θ_{depth}	θ_{vars}	Learn	1	2	3	4	5	avg
2	1	DF	89.2	90.6	90.1	95.2	88.0	90.6
2	1	k-NN$_{>1}$	89.2	90.6	90.1	95.2	88.0	90.6
3	2	DF	89.2	90.6	90.8	95.2	88.0	90.8
3	2	k-NN$_{>1}$	89.2	90.6	90.3	95.2	88.0	90.7
4	3	DF	79.2	90.6	58.9	89.6	88.0	81.3
4	3	k-NN$_5$	88.0	90.6	70.9	88.8	88.0	85.3
4	3	k-NN$_{10}$	89.4	95.7	78.6	94.2	81.1	87.8
4	3	k-NN$_{100}$	86.6	95.7	90.8	95.8	81.1	90.0
hand		DF	89.2	90.6	90.8	95.2	88.0	90.8
hand		k-NN$_5$	93.6	99.2	96.8	96.6	91.3	95.5
hand		k-NN$_{10}$	94.2	98.9	97.1	97.7	94.5	96.5
hand		k-NN$_{100}$	94.7	99.7	98.0	99.3	95.3	97.4
hand		k-NN$_{500}$	97.6	99.7	98.0	99.3	95.3	98.0

Table 9
Area under the recall-precision curve (AURPC) results on the CORA dataset. Horizontal lines separate different learning parameter settings (allowing for increasingly complex TETs to be learned), plus a manually curated TET (hand). Rows within each learning setting indicate a different predictive model, either discriminant function (DF) or k-NN with the TET metric, for different values of $k > 1$ (only values producing non-negligible differences are reported). Columns report results for each fold and macro-averaged over the five folds.

θ_{depth}	θ_{vars}	Learn	1	2	3	4	5	avg
2	1	DF	0.926	0.993	0.940	0.980	0.957	0.959
2	1	k-NN$_{>1}$	0.966	0.993	0.940	0.980	0.957	0.967
3	2	DF	0.926	0.993	0.946	0.980	0.957	0.960
3	2	k-NN$_{>1}$	0.966	0.993	0.956	0.980	0.957	0.970
4	3	DF	0.872	0.993	0.986	0.960	0.957	0.954
4	3	k-NN$_5$	0.906	0.993	0.880	0.925	0.957	0.932
4	3	k-NN$_{10}$	0.932	0.993	0.928	0.969	0.957	0.956
4	3	k-NN$_{100}$	0.945	0.993	0.960	0.982	0.957	0.967
hand		DF	0.926	0.993	0.946	0.980	0.957	0.960
hand		k-NN$_5$	0.992	1.00	0.986	0.998	0.962	0.988
hand		k-NN$_{10}$	0.992	1.00	0.995	0.999	0.962	0.990
hand		k-NN$_{100}$	0.994	1.00	0.996	1.00	0.953	0.989
hand		k-NN$_{500}$	0.995	1.00	0.992	1.00	0.971	0.992

the k-NN algorithm based on the complete learned TET performs slightly better than the simple TET for longer prediction horizons. This happens because the TET metric, although taking into account count-of-counts features (and thus performing consistently better than plain counts-based metrics), is not able to exactly compute the h-index starting from these features.

6.2. Cora

CORA is a dataset of research papers and their citations, originally collected by Andrew McCallum and used for different predictive tasks including hierarchical classification, information extraction, and citation matching. Here we focus on this latter task, namely predicting whether two bibliographic records refer to the same paper. We rely on the relational data representation and experimental setting defined by Singla and Domingos [40].

The domain consists of entities of types *title*, *author*, *venue* (given by a string value), *bibrec* (a bibliographic record given by its author, title and venue fields), as well as *title_word*, *author_word*, *venue_word*, which are the constituent words appearing in *title*, *author*, and *venue*. Only the first author of each record is considered in this setting. The data set contains 1295 bibliographic records, referring to 132 different research papers, 50 authors, and 103 venues. Relations are all representing part-of relationships: *title_of*, *author_of*, and *venue_of* link bibliographic records to their constituent fields; *word_in_title*, *word_in_author*, *word_in_venue* link complete field strings to their constituent words. Note that relations in the first group are one-to-one, whereas the second group is one-to-many.

The experimental setting by Singla and Domingos [40] consists of a five fold cross validation procedure over *plausible* candidate pairs as identified using McCallum et al.'s canopy approach [28], with TF-IDF cosine as the similarity measure. This results in 52,923 overall candidate pairs, with 30,971 positive and 21,952 negative pairs respectively. The compiled dataset is available at `alchemy.cs.washington.edu`.

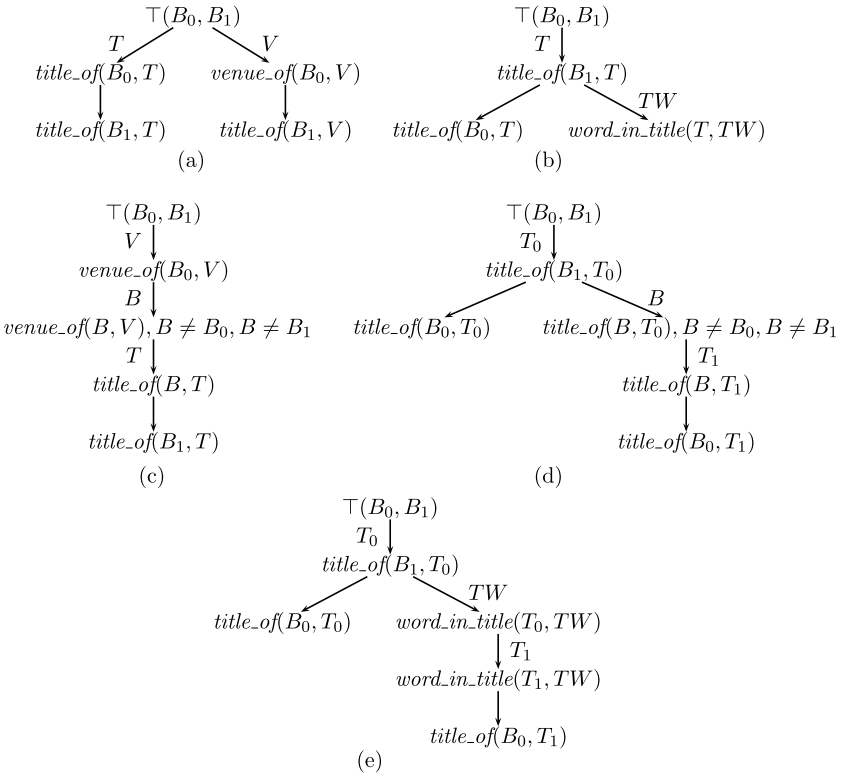


Fig. 7. Features in the CORA TETs for entity resolution on bibliographic records.

We fixed $\theta_{RIG} = 0$, as in the DBLP experiments, and $\theta_{score} = 1e-2$. We ran the TET learner on each of the five different training sets with increasing values of the main parameters controlling the size of the search space, θ_{depth} and θ_{vars}. Tables 8 and 9 report F1 and Area under the recall-precision curve (AURPC) values for each fold and macro-averaged on the five folds. The simple discriminant function guiding the TET learning phase (DF) is compared to k-NN with the TET metric, for different values of $k > 1$ (only values producing non-negligible differences are reported). We did not perform any fine tuning on the TET metric, leaving all hyper-parameters to one (see Section 4.1).

The first apparent finding is that small TETs seem to perform quite well on this dataset, and increasing their complexity does not pay much. Fig. 7(a) shows the TET learned in four (1, 2, 4, 5) out of five folds for both ($\theta_{depth} = 2$, $\theta_{vars} = 1$) and ($\theta_{depth} = 3$, $\theta_{vars} = 2$) parameter settings and in two (2, 5) folds for the ($\theta_{depth} = 4$, $\theta_{vars} = 3$) setting. This represents the very basic feature of the two records B_0, B_1 having identical title and/or venue fields. It is noteworthy that neither here, nor in any of the other TETs, features involving the author field were constructed. This may be due to the fact that the CORA dataset contains relatively many different publications by a relatively small number of (first) authors, so that the author field becomes a quite poor predictor for the identity of papers. Note that simple pairs with clearly different first authors were preliminarily excluded by the canopy construction [28] and are not part of the plausible candidates. The only difference between the two simpler learning settings is in fold three, where the ($\theta_{depth} = 3$, $\theta_{vars} = 2$) setting learns the TET in Fig. 7(b), while the TET learned by the simplest setting ($\theta_{depth} = 2$, $\theta_{vars} = 1$) lacks the $\xrightarrow{TW}$ *word_in_title*(T, TW) branch. The former achieves slightly better results, possibly because of a correlation between the number of title words and the likelihood that two entries refer to the same paper.

Learning more complex TETs does not seem to provide improvements in this setting. However, an inspection on the learned TETs gives interesting insights on the potential of the mined features. Figs. 7(c) and (d) show linear branches that constitute the main features in two of the complex TETs. (c), at first, appears to be a fairly complex feature, which because of its three variable introductions would represent a three-level hierarchical count. However, due to the one-to-one nature of the relations in the extensions $\xrightarrow{V}$ *venue_of*(B_0, V) and $\xrightarrow{T}$ *title_of*(B, T), only the extension $\xrightarrow{B}$ *venue_of*(B, V) introduces real counts other than 0 and 1. Roughly speaking, the feature (c) counts the number of records B that have the same venue as B_0, and the same title as B_1. In both TETs containing the branch (c), also the dual branch with the roles of B_0 and B_1 interchanged was constructed. Intuitively, this feature uses the transitivity of the *same_paper* relation by considering "interpolating" records B for which there is evidence that they are equal to B_0 because of agreement in the venue field, and equal to B_1 because of agreement in the title field (or vice-versa). Note that the dataset includes 103 distinct venues for the 132 papers, which makes venue almost as discriminative as title.

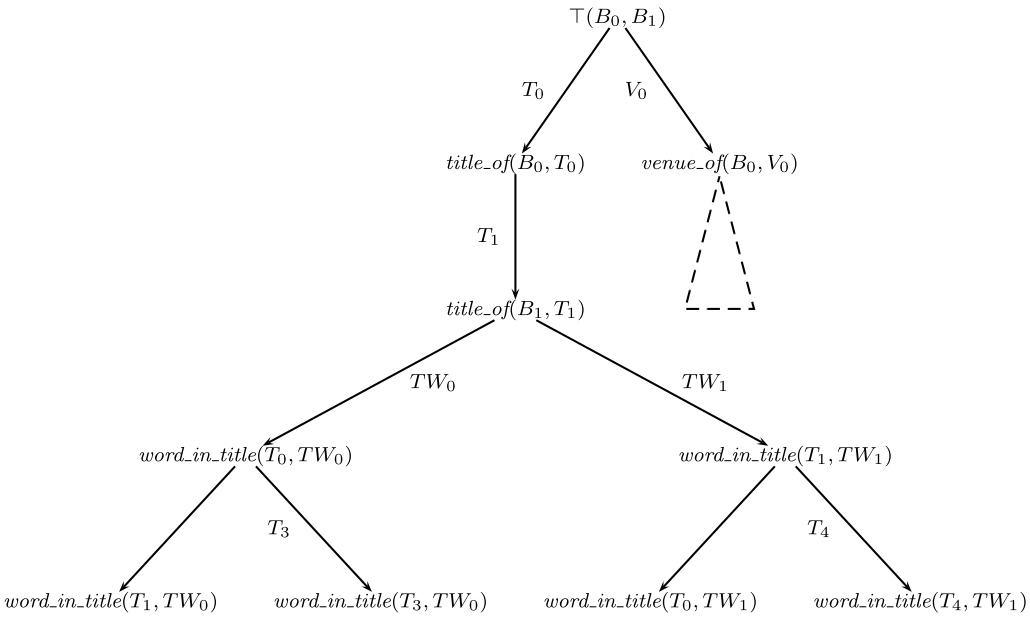


Fig. 8. Handcrafted CORA TET representing *idf-like* counts-of-counts features. The dashed triangle indicates a copy of the *title* subtree, with *title* replaced by *venue*.

(d) is a refinement of the left branch of (a): in addition to testing equality of the title field, (d) also counts the number of additional records B that have the same title. It appears, however, that the introduction of the variable T_1 in the extension $\xrightarrow{T_1} title\_of(B, T_1)$ here is redundant, since T_0 at this place is already established as the unique title of B, and so the simpler extension $\rightarrow title\_of(B_0, T_0)$ would express the same logical feature. The reason for the roundabout way (d) ends up taking for defining this feature lies in the fact that our learning algorithm includes a constraint that variables introduced at a previous extension (B in this case) should be used in the next child node (see Section 5).

With the exception of the $\xrightarrow{TW} word\_in\_title(T, TW)$ branch of TET (b), all features discussed so far do not use the word-related relations. Comparisons are based on identity of whole title and venue strings. This is different in (e), which shows the complete TET learned for the third fold in the ($\theta_{depth} = 4$, $\theta_{vars} = 3$) learning setting. It again is a refinement of the same-title feature, but this time in its right sub-branch also introducing counts of title words, and by means of the subsequent extension $\xrightarrow{T_1} word\_in\_title(T_1, TW) \rightarrow title\_of(B_0, T_1)$ a count-of-count feature that incorporates an inverse-document frequency feature (cf. TET (2) in Example 2.4). To understand the meaning of this branch, consider how the sub-TET rooted at $word\_in\_title(T_0, TW)$ evaluates for a title word w: if w does not occur in the title of B_1, then the value is f. If w is in the title of B_1, but not in the title of B_0, then the value is $(t, \{(t, \{f\}) : k_1, f : k_2\})$, where k_1 and k_2 are the number of titles in the domain that contain, respectively do not contain, w. If, finally, w occurs both in the title of B_1 and B_0, then the value is $(t, \{(t, \{t\}) : k_1, f : k_2\})$, with the same k_1, k_2 as before. The branch $\xrightarrow{TW} \cdots$ then provides for each of these possible values the count of words w with that value. In this manner, the values of the right branch provide all the count-of-count statistics required for the inverse-document-frequency feature. Note that the more parsimonious representation (2) is not learnable by our current TET learner, since we only allow single-literal nodes.

We see, thus, that the learned complex TETs represent very reasonable features. In order to better understand whether the lack of performance gain over the simpler TETs is possibly due to lack of predictive relevance of these features, or due to a difficulty for the TET metric to utilize the feature information in the way it is presented by these TETs, we manually designed a TET that encodes the *idf-like* feature represented by (e) in a way that is optimized for the TET metric (Fig. 8). It provides separate branches for words in each of the titles and each of the venues (the dashed triangle indicates a copy of the *title* subtree, with *title* replaced by *venue*). As shown in the last rows of Tables 8 and 9, the TET achieves quite high performance when paired with k-NN employing the TET-metric, with almost perfect AURPC in most folds, while the discriminant function again fails to exploit the full potential of counts-of-counts features.

We can conclude that the TET learner is able to discover complex features with high discriminative value. The fact that some post-processing in the representation of the features was needed to obtain the best performance results with k-NN prediction may not be very surprising, since the learner is not optimizing with regard to the TET metric. In future work this kind of post-processing may be automated by implementing efficient techniques for optimizing the parameters of the TET metric (weight parameters ω and normalization labels y), and optimizing the TET structure using post-pruning and -balancing operations.

The results of the learned TETs are comparable with those achieved with a Markov Logic Network (MLN) that (like our TET) is language independent, i.e. does not contain rules referring to specific strings occurring in the data, which achieves

an AURPC of 0.971 [40]. Note that the MLN based approach in [40] – as well as more recent approaches achieving still higher accuracy [33,39] – perform collective classification, and therefore can exploit the fact that the binary relation on bibliographic records that one predicts is an equivalence relation. The two classification models we have used both perform independent predictions for each pair of bibliographic records, and therefore cannot be expected to achieve results that are competitive with state-of-the-art collective approaches. It should be emphasized, though, that in [33,39] the MLN structure (i.e. the set of logical formulas) was carefully designed by hand, while in our experiments the TET structure is learned from data. A carefully crafted TET as described in Fig. 8 indeed achieves a macro-averaged AURPC of around 0.99. Additionally, TET features could equally well be used in connection with collective classification techniques. We also compared TET results with those achievable by TILDE, with and without aggregates. As for the DBLP case, the search procedure of TILDE suffers from a lack of direct informativeness of single predicates, and plain TILDE returns an empty tree for all folds. However, exhaustive lookahead allows us to overcome the problem and recover the same rules of the simple TET in Fig. 7(a), thus achieving substantially equivalent results (macro averaged F1 = 91.0%, slightly better than those of the simple TET as Tilde learns these rules for the third fold too). More complex features, like the *idf-like* ones in Figs. 7(e) and 8 cannot be recovered by plain Tilde, and adding aggregates in the language bias concerning counts of author, title and venue words dramatically increases learning time: the search did not finish after a week of CPU time.

As for inductive logic programming and relational rule learning approaches, learning time strongly depends on the constraints imposed on the search space, which is otherwise exponential in the number of candidate predicates. Learning TETs takes roughly three minutes, one hour and 20 hours respectively, on average over the five folds, for the three increasingly complex learning settings ($\theta_{depth} = 2$, $\theta_{vars} = 1$), ($\theta_{depth} = 3$, $\theta_{vars} = 2$), ($\theta_{depth} = 4$, $\theta_{vars} = 3$). As a matter of comparison, Tilde learns the tree resembling TET in Fig. 7(a) in 30 seconds or 11 hours, depending on the number of exhaustive lookaheads allowed (one and two, respectively).

7. Conclusion

Properties of entities in a relational domain can depend on complex combinatorial count-of-count features characterizing the entities' relational neighborhood. Examples of properties that are directly defined in terms of count-of-count features are the h-index of an author, and certain relevance measures widely used in information retrieval. Type Extension Trees are a simple, but highly expressive representation language for count-of-count features. In this article we have presented a method for learning Type Extension Trees in supervised learning settings as a means of discovering count-of-count features that are informative for the prediction of a class label.

Most existing frameworks for statistical relational learning either are only based on simpler, "flat", count features, or their use of count-of-count features is only implicit in the specification of conditional probability distributions, and does not include an interpretable representation of the underlying features. Examples of frameworks of the first kind are Markov Logic Networks [36] (cf. Example 2.10), and systems providing simple aggregation operators [1,13,18]. Examples of frameworks of the second kind are probabilistic relational models that allow the specification of conditional probability distribution using nested combination functions [16,30].

Kernel methods can be also applied to (implicitly) extract features from relational data. The general framework of convolution kernels [12] has originated a wealth of different approaches for defining the similarity between structured objects (see e.g. [45] and references therein). Features defined by these kernels essentially count fragments or substructures, but not counts of counts. In most cases, these methods aim to develop a suitable representation of structured data for subsequent learning, not to discover features. There are previous works, however, where the feature space itself is learned from relational data [23,29] and is interpretable in terms of definite clauses.

In most of these previous works relational feature construction is an integral part of a particular learning paradigm. Relational features in their own right have previously been investigated in [32]. Here a systematic view of aggregation-based features at different levels of complexity is developed. However, the focus still is on aggregation over a single level of relational dependencies.

Discovered TET features can be used in a variety of classification models, and could be integrated into existing models such as relational probability trees [31], or inductive logic programming systems, for which simpler types of count features have already been used [1]. In this paper we have considered two approaches for directly augmenting TET features into full prediction models. The simple discriminant function is fast to learn and evaluate, but only makes limited use of the count-of-count information provided by a TET feature value. We have therefore also introduced a metric on TET values defined by a recursive application of the Wasserstein–Kantorovich metric. With this metric, distance-based methods for supervised or unsupervised learning become directly applicable.

Our experiments have shown that our TET learning algorithm is able to discover non-trivial and interpretable count-of-count features. A comparison of the classification accuracies achieved with the discriminant function model and k-nearest neighbor classification based on the TET metric indicates that TET features learned using the discriminant function can also support other classification models, and that a model that exploits the complex count-of-count information outperforms models only using flat counts.

8. Supplementary material

The software for TET learning and for the computation of Wasserstein–Kantorovich metric between TET values, together with the data used in the experiments presented in this paper, can be downloaded at http://www3.diism.unisi.it/~lippi/research/TET.html.

Acknowledgements

AP, ML and PF were partially supported by PRIN grant 2009LNP494_002.

References

[1] A. Van Assche, C. Vens, H. Blockeel, S. Dzeroski, First order random forests: Learning relational classifiers with complex aggregates, Mach. Learn. 64 (2006) 149–182.
[2] Carl Bergstrom, Measuring the value and prestige of scholarly journals, Coll. Res. Libr. News 68 (5) (2007) 314–316.
[3] H. Blockeel, L. De Raedt, Top-down induction of first-order logical decision trees, Artif. Intell. 101 (1–2) (1998) 285–297.
[4] Vladimir I. Bogachev, Aleksandr V. Kolesnikov, The Monge–Kantorovich problem: achievements, connections, and perspectives, Russ. Math. Surv. 67 (5) (2012) 785.
[5] Sergey Brin, Lawrence Page, The anatomy of a large-scale hypertextual web search engine, Comput. Netw. 30 (1–7) (1998) 107–117.
[6] Luc De Raedt, Paolo Frasconi, Kristian Kersting, Stephen Muggleton (Eds.), Probabilistic Inductive Logic Programming: Theory and Applications, Lecture Notes in Computer Science, vol. 4911, Springer, Berlin, 2008.
[7] P. Frasconi, M. Jaeger, A. Passerini, Feature discovery with type extension trees, in: Proceedings of the 18th Int. Conf. on Inductive Logic Programming (ILP), in: Lecture Notes in Artificial Intelligence, vol. 5194, 2008, pp. 122–139.
[8] N. Friedman, Lise Getoor, D. Koller, A. Pfeffer, Learning probabilistic relational models, in: Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99), 1999.
[9] T. Gärtner, P. Flach, S. Wrobel, On graph kernels: Hardness results and efficient alternatives, in: Proceedings of the 6th Annual Conference on Computational Learning Theory and the 7th Kernel Workshop, in: LNAI, vol. 2777, 2003, pp. 129–143.
[10] Lise Getoor, Ben Taskar (Eds.), Introduction to Statistical Relational Learning, MIT Press, Cambridge, MA, 2007.
[11] E. Grädel, Finite model theory and descriptive complexity, in: Finite Model Theory and Its Applications, in: Texts in Theoretical Computer Science, Springer, 2007, Chapter 3.
[12] D. Haussler, Convolution kernels on discrete structures, Technical report 99-10, UCSC-CRL, 1999.
[13] D. Heckerman, C. Meek, D. Koller, Probabilistic entity-relationship models, PRMs, and plate models, in: L. Getoor, B. Taskar (Eds.), Introduction to Statistical Relational Learning, MIT Press, 2007.
[14] J.E. Hirsch, An index to quantify an individual's scientific research output, Proc. Natl. Acad. Sci. 102 (46) (2005) 16569–16572.
[15] Wilfrid Hodges, Model Theory, Cambridge University Press, 1993.
[16] M. Jaeger, Relational Bayesian networks, in: Proceedings of the 13th Conference of Uncertainty in Artificial Intelligence (UAI-13), Providence, USA, Morgan Kaufmann, 1997, pp. 266–273.
[17] M. Jaeger, Type extension trees: A unified framework for relational feature construction, in: Proceedings of Mining and Learning with Graphs (MLG-06), 2006.
[18] Y. Kavurucu, P. Senkul, I.H. Toroslu, Concept discovery on relational databases: New techniques for search space pruning and rule quality improvement, Knowl.-Based Syst. 23 (8) (2010) 743–756.
[19] A. Knobbe, M. de Haas, A. Siebes, Propositionalisation and aggregates, in: Proceedings of PKDD 2001, 2001, pp. 277–288.
[20] A.J. Knobbe, A. Siebes, D. van der Wallen, Multi-relational decision tree induction, in: Proceedings of PKDD-99, 1999, pp. 378–383.
[21] Stanley Kok, Pedro Domingos, Learning the structure of Markov logic networks, in: Luc De Raedt, Stefan Wrobel (Eds.), ICML, in: ACM International Conference Proceeding Series, ACM, 2005, pp. 441–448.
[22] M.-A. Krogel, S. Wrobel, Transformation-based learning using multirelational aggregation, in: Proceedings of ILP 2001, in: LNAI, vol. 2157, 2001, pp. 142–155.
[23] N. Landwehr, A. Passerini, L. De Raedt, P. Frasconi, Fast learning of relational kernels, Mach. Learn. 78 (3) (2010) 305–342.
[24] Michael Ley, The dblp computer science bibliography: Evolution, research issues, perspectives, in: Alberto H.F. Laender, Arlindo L. Oliveira (Eds.), SPIRE, in: Lecture Notes in Computer Science, vol. 2476, Springer, 2002, pp. 1–10.
[25] Haibin Ling, Kazunori Okada, An efficient earth mover's distance algorithm for robust histogram comparison, IEEE Trans. Pattern Anal. Mach. Intell. 29 (5) (2007) 840–853.
[26] Marco Lippi, Manfred Jaeger, Paolo Frasconi, Andrea Passerini, Relational information gain, Mach. Learn. 83 (2) (2011) 219–239.
[27] Nathan N. Liu, Qiang Yang, Eigenrank: a ranking-oriented approach to collaborative filtering, in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2008, pp. 83–90.
[28] Andrew McCallum, Kamal Nigam, Lyle H. Ungar, Efficient clustering of high-dimensional data sets with application to reference matching, in: Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '00, New York, NY, USA, ACM, 2000, pp. 169–178.
[29] S. Muggleton, H. Lodhi, A. Amini, M. Sternberg, Support vector inductive logic programming, Innov. Mach. Learn. (2006) 113–135.
[30] S. Natarajan, P. Tadepalli, E. Altendorf, T.G. Dietterich, A. Fern, A. Restificar, Learning first-order probabilistic models with combining rules, in: Proceedings of the 22nd International Conference on Machine Learning (ICML-05), 2005, pp. 609–616.
[31] Jennifer Neville, David Jensen, Lisa Friedland, Michael Hay, Learning relational probability trees, in: Lise Getoor, Ted E. Senator, Pedro Domingos, Christos Faloutsos (Eds.), KDD, ACM, 2003, pp. 625–630.
[32] C. Perlich, F. Provost, Aggregation-based feature invention and relational concept classes, in: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003), 2003.
[33] Hoifung Poon, Pedro Domingos, Joint inference in information extraction, in: Proceedings of AAAI-07, 2007, pp. 913–918.
[34] J.R. Quinlan, Learning logical definitions from relations, Mach. Learn. 5 (1990) 239–266.
[35] L. De Raedt, Logical and Relational Learning, Springer, 2008.
[36] M. Richardson, P. Domingos, Markov logic networks, Mach. Learn. 62 (1–2) (2006) 107–136.
[37] Yossi Rubner, Carlo Tomasi, Leonidas J. Guibas, The earth mover's distance as a metric for image retrieval, Int. J. Comput. Vis. 40 (2) (2000) 99–121.
[38] Ajit Paul Singh, Geoffrey J. Gordon, Relational learning via collective matrix factorization, in: Proceedings of KDD-08, 2008, pp. 650–658.

[39] Sameer Singh, Karl Schultz, Andrew Mccallum, Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs, in: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II, ECML PKDD '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 414–429.
[40] P. Singla, P. Domingos, Entity resolution with Markov logic, in: Proceedings of ICDM-06, 2006.
[41] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, Zhong Su, Arnetminer: extraction and mining of academic social networks, in: Proceedings of KDD-08, 2008, pp. 990–998.
[42] Ben Taskar, Carlos Guestrin, Daphne Koller, Max-margin Markov networks, in: Sebastian Thrun, Lawrence Saul, Bernhard Schölkopf (Eds.), Advances in Neural Information Processing Systems 16, MIT Press, Cambridge, MA, 2004.
[43] G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, L. De Raedt, Lifted probabilistic inference by first-order knowledge compilation, in: Proceedings of IJCAI 2011, 2011.
[44] C. Vens, J. Ramon, H. Blockeel, Refining aggregate conditions in relational learning, in: Proceedings of PKDD 2006, in: LNAI, vol. 4213, 2006, pp. 383–394.
[45] S.V.N. Vishwanathan, N.N. Schraudolph, R. Kondor, K.M. Borgwardt, Graph kernels, J. Mach. Learn. Res. 99 (2010) 1201–1242.