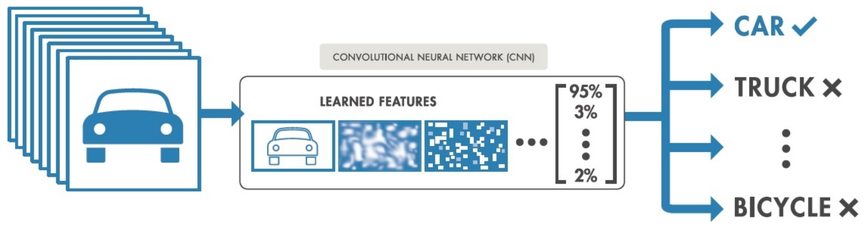
Thinking Logically about Deep Learning

Sebastijan Dumančić, Tias Guns, Wannes Meert, Hendrik Blockeel
{firstname,lastname}@cs.kuleuven.be, tias.guns@vub.be
KU Leuven, VUB, Belgium

Why should you care?

Deep learning revolution

- addresses the problem of learning in high-dimensional spaces
- improved state-of-the-art in many applications



Core idea:
- learn data representation simultaneously with a model
- *simplify* the learning task first

Benefit for Deep Learning:
- more expressivity and broader applicability
- better understanding

Statistical Relational Learning and Probabilistic ILP

- combine logical reasoning with probability theory
- can reason about complex data structures with uncertainty

```
0.1 : :burglary.    0.7 : :hears_alarm(mary).
0.2 : :earthquake.  0.4 : :hears_alarm(john).
alarm :- earthquake.
alarm :- burglary.
calls(X) :- alarm, hears_alarm(X).
call :- calls(X).
```

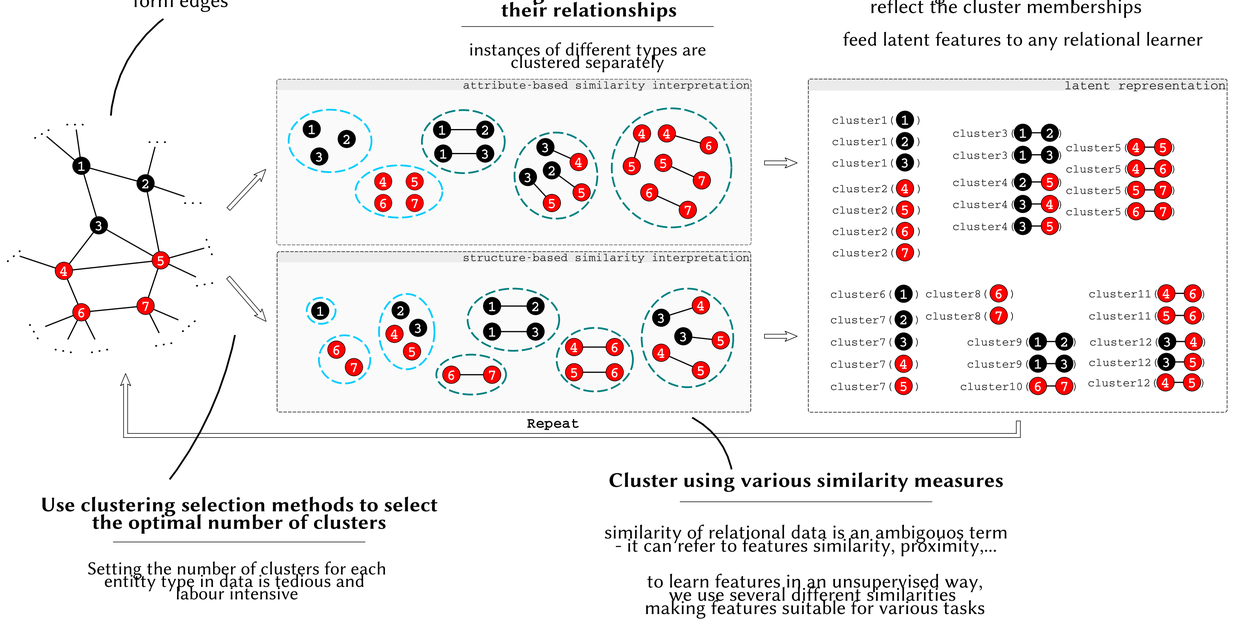
Problem:
- Learning and inference suffer from combinatorial explosion
- *high-dimensional* in terms of possible logical formulas

Can deep learning help simplify learning in SRL/PILP models?

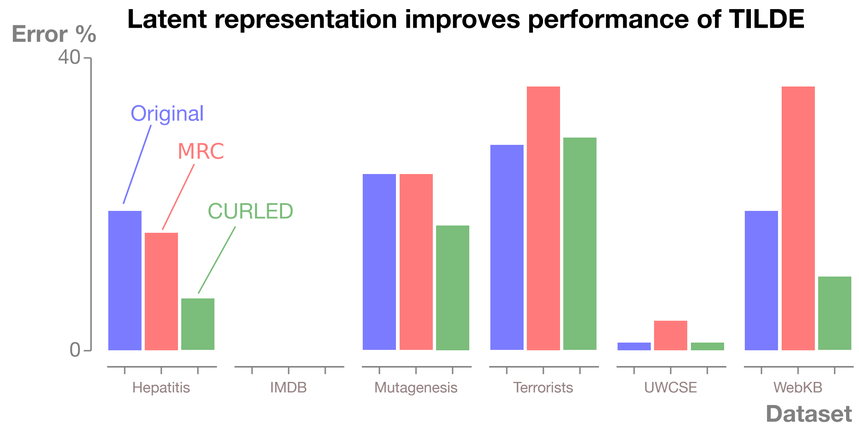
Requirement from SRL/PILP:
- needs to work with predicate logic

What have we done so far?

Using approximate symmetries to define latent features and relations (CUR^{2}LED)



Core idea: learn a distributed representation of data by alternative what does an *approximate symmetry* means (and rely on clustering to detect it)



Relational auto-encoders

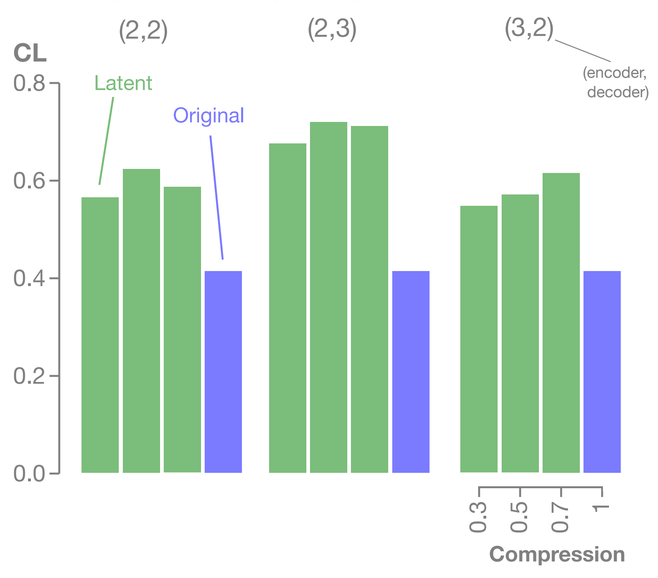
Auto-encoders are **extremely versatile** deep learning component:
- they are based on a very simple principle of *learning representation by compression*:
- applicable as a component in un/semi/supervised learning setup
- *suitable for the variety of tasks and inference types in SRL/PILP*

Goal: implement auto-encoders as *logic programs*

```
Input: mother(anna,dirk).  female(anna).  father(tom,dirk).  male(tom).
Encoder:                          latent1(X,Y) :- mother(X,Y);father(X,Y).
                                  latent2(X) :- female(X).
Latent rep.:      latent1(anna,dirk).  latent1(tom,dirk).  latent2(anna).
Decoder:                          mother(X,Y) :- latent1(X,Y),latent2(X).
                                  female(X) :- latent2(X).
                        father(X,Y) :- latent1(X,Y),not(latent2(X)).
                                  male(X) :- not(latent2(X)).
Output: mother(anna,dirk).  female(anna).  father(tom,dirk).  male(tom).
```

Components of the framework

- **Language bias:** declarative specification how to construct encoder and decoder clauses (provided by the user)
- **Latent properties:** a declarative specification of properties which latent representation should poses
- **Mapper:** a component that takes the data, language bias and latent properties, and formulates the learning as a *constraint satisfaction problem*
- **Solver:** constraint programming, weighted MaxSAT, ...



LaTeX TikZposter

Thinking Logically about Deep Learning

Sebastijan Dumančić
sebastijan.dumancic@cs.kuleuven.be
https://people.cs.kuleuven.be/~sebastijan.dumancic
KU Leuven, Belgium

References

S. Dumančić and H. Blockeel: *An expressive dissimilarity measure for relational clustering using neighbourhood trees*, MLJ 2017

S. Dumančić and H. Blockeel: *Clustering-Based Unsupervised Relational Representation Learning with Explicit Distributed Representation*, IJCAI 2017

S. Dumančić and H. Blockeel: *Demystifying Relational Latent Representations*, ILP 2017

S. Dumančić, T. Guns, W. Meert, H. Blockeel: *Relational Auto-encoders*, soon to be on ArXiv

Code

ReCeNT
A Scala toolbox for distance-based relational learning and clustering. Implements many distance measures for relational data.
Available on Github: https://github.com/sebdumancic/ReCeNT

CUR^{2}LED
A Scala toolbox for learning relational latent feature hierarchies with clustering and explicitly defined distributed representation.
Available on Github: https://github.com/sebdumancic/CURLED

What else exists?

If you are interested in an overview covering **intersection between (statistical) relational and deep learning**, I am actively maintaining the website surveying the existing approaches:https://goo.gl/KjUPMU

Booklet

Get the poster and the papers in one place: https://goo.gl/n1aQ7A

If you are interested and active semi-supervised clustering

My colleague Toon Van Craenendonck and I have recently introduce a novel approach to semi-supervised clustering, named COBRA. It is based on the idea of *lifting small compact region in feature space*, and merging those regions based on pairwise constraints.

T. Van Craenendonck, S. Dumančić and H. Blockeel: *COBRA: A Fast and Simple Method for Active Clustering with Pairwise Constraints*, IJCAI 2017

Clustering-Based Relational Unsupervised Representation Learning with an Explicit Distributed Representation

Sebastijan Dumančić and **Hendrik Blockeel**
Computer Science Department, KU Leuven, Belgium
{sebastijan.dumancic,hendrik.blockeel}@cs.kuleuven.be

Abstract

The goal of unsupervised representation learning is to extract a new representation of data, such that solving many different tasks becomes easier. Existing methods typically focus on vectorized data and offer little support for relational data, which additionally describe relationships among instances. In this work we introduce an approach for *relational* unsupervised representation learning. Viewing a relational dataset as a hypergraph, new features are obtained by clustering vertices and hyperedges. To find a representation suited for many relational learning tasks, a wide range of similarities between relational objects is considered, e.g. feature and structural similarities. We experimentally evaluate the proposed approach and show that models learned on such latent representations perform better, have lower complexity, and outperform the existing approaches on classification tasks.

1 Introduction

Every machine learning task inherently depends on the quality of provided features. A good set of features is thus a crucial precondition for the good performance of any classifier. Yet, finding such a set in practice has proven to be a tedious and time-consuming task. Furthermore, substantial domain knowledge and exploration are often required. To address this issue, *representation learning* [Bengio, 2009] focuses on automatic learning of good multi-level data representations.

Representation learning methods include two categories. Supervised representation learning learns a hierarchy of new features in a discriminative way. Thus, the representation is tailored specifically for a given task. An example of such an approach are convolutional neural networks. In contrast, the unsupervised representation learning ($\mathcal{URL}$) [Hinton and Salakhutdinov, 2006; Bengio *et al.*, 2007; Ranzato *et al.*, 2007] takes a *generative stance*. Because it requires no supervision, such representation could be shared among multiple tasks. This is the direction we explore. Intuitively, these methods find useful features by compressing the original data by means of a multiple-clustering procedure, in which an instance can belong to more than one cluster. The features obtained by $\mathcal{URL}$ typically indicate cluster assignments of each instance. Consequently, a classifier learns from cluster memberships instead of the original data.

One major limitation of the existing methods is that they focus on vectorized data representations. Hence, the ubiquitous and abundant structured and relational data are not well supported. In contrast to the vectorized representations, relational data describe instances together with their relationships. This is often viewed as a hypergraph[1], in which instances form vertices and their relationships form hyperedges. Such data emerges in many real-life problems. For instance, chemical and biological data describing molecules or protein interaction networks are naturally expressed in graph-structured formats. Another example includes social networks, where many instances interact with each other. A common language for expressing such data is *predicate logic*, which further subsumes any data stored in a relational database.

This work focuses on unsupervised representation learning with relational data. To this end, we introduce CUR^{2}LED - a *clustering-based unsupervised relational representation learning with explicit distributed representation*. CUR^{2}LED is inspired by the work of Coates *et al.*, 2011, in which the authors introduce a general pipeline for learning a feature hierarchy by means of clustering. Assuming a spatial order of features (i.e., pixels), the introduced framework (i) extracts image patches, i.e., subsets of pixel from the original images candidating as a potential high-level feature, (ii) pre-processes each patch (e.g. normalization and whitening), and (iii) learns a feature-mapping by clustering image patches. The authors show that such general procedure with a simple k-means algorithm can perform as well as the specialized algorithms, such as auto-encoders and Restricted Boltzmann machines.

CUR^{2}LED learns a new representation by clustering both instances and their relationships. What is distinctive about CUR^{2}LED is that the relational structure is preserved throughout the hierarchy, contrasted to the existing approaches that map relational structures onto a vectorized representation. Another distinctive feature of CUR^{2}LED is the notion of *similarity interpretation*. When clustering relational data, a similarity of relational objects is an ambiguous concept. Two relational objects might be similar according to their attributes, relationships, or a combination of both. The notion of similarity

[1] A hypergraph is a graph in which edges can connect more than two vertices.

interpretation precisely states the exact source of similarity used. CUR^{2}LED exploits this ambiguity to its advantage by using the similarity interpretations to encode a *distributed representation* of data.

Distributed representation is one of the pillars underlying the success of representation learning methods. Intuitively, it refers to a concept of reasonably-sized representation that captures a huge number of possible configurations [Bengio *et al.*, 2013]. In contrast to the one-hot representations which require N parameters to represent N regions, distributed representations require N parameters to represent up to 2^N regions. The main difference is that a concept within a distributed representation is represented with several independently manipulated factors, instead of exactly one factor as with one-hot representations. Thus, such representations are substantially more expressive. In that sense, the similarity interpretation defines the exact factors that can be manipulated individually to represent individual concepts.

The contributions of this paper include (i) a general framework for learning relational feature hierarchies by means of clustering, (ii) a principled way of generating distributed relational representations based on different similarity interpretations, (iii) a general framework for hyperedge clustering and (iv) the experimental evaluation of the proposed framework.

In the following section, we briefly review related work. Next, we outline our approach and discuss arising issues in Section 3. We then briefly present the similarity measure used for clustering relational data, discuss its extension towards hyperedge clustering, and formally define the notion of similarity interpretation. Experimental results are discussed in Section 5.

2 Related work

Clustering has been previously recognized as an effective way of enhancing relational learners. Popescul and Ungar (2004) apply k-means clustering to the objects of each type in a domain, create predicates for new clusters and add them to the original data. *Multiple relational clustering (MRC)* [Kok and Domingos, 2007; 2008] is a relational probabilistic clustering framework based on Markov logic networks [Richardson and Domingos, 2006] clustering both vertices and relationships. Both approaches are instances of predicate invention tasks[Kramer, 1995; Craven and Slattery, 2001], concerned with extending the vocabulary given to a learner by discovering novel concepts in data. CUR^{2}LED differs in several ways. Whereas Popescul and Ungar develop a method specifically for document classification, CUR^{2}LED is a general *off-the-shelf* procedure that can be applied to any relational domain. Moreover, CUR^{2}LED clusters both instances and relations, whereas Popescul and Ungar cluster only instances. In contrast to MRC which does not put any assumptions in the model, CUR^{2}LED is a more informed approach that explicitly defines different notions of relational similarity to be used for clustering. Though MRC was used as a component in structure learning, it does not provide new language constructs, but simplifies the search over possible formulas. CUR^{2}LED learns a model directly from the new features.

Much of the recent work focused on constructing the *embeddings* of relational constructs [Niepert, 2016; Bordes *et al.*, 2011; Yang *et al.*, 2014; Bordes *et al.*, 2013; Henaff *et al.*, 2015; Niepert *et al.*, 2016]. This work maps relational concepts to low dimensional vector spaces, and therefore loses the relationship information. Moreover, it focuses on supervised learning. Thus, CUR^{2}LED differs in that it learns a *relational* feature hierarchy in an unsupervised manner.

3 Representation learning via clustering

Several issues arise from devising a general relational feature hierarchy pipeline, all of them due to the complexity of relational data. Firstly, the issue is *what should be clustered?* In the i.i.d. case (drawn independently from the same population), the dataset contains only instances and their features, thus, one clusters the instances. However, relational data additionally describes relationships among instances. Furthermore, it can vary from a single large network of many interconnected entities (a *mega-example*), to a set of many disconnected networks where each network is an example. Ideally, one would address both cases. CUR^{2}LED assumes that relational data is provided as a labelled hypergraph, where examples form vertices and relations between them form hyperedges, and does not make a distinction between the above-mentioned cases. Formally said, the data structure is a typed, labelled hypergraph $H = (V, E, \tau, \lambda)$ with V being a set of vertices, E a set of hyperedges, τ a function assigning a type to each vertex and hyperedge, and λ a function assigning a vector of values to each vertex.

CUR^{2}LED learns a new representation by clustering both *vertices and hyperedges in a hypergraph*. Considering that vertices have associated types, CUR^{2}LED does not allow mixing of types, i.e., a cluster can contain only vertices of the same type. The same holds for hyperedges which connect vertices of different types.

The second issue emerges with *the ambiguity of similarity in relational context.* With the i.i.d. data, the features are the only source of similarity between examples. In the relational context, a similarity can originate in the features of relational objects, structures of their neighbourhoods (both features- and relationship-wise), interconnectivity or graph proximity, just to name a few. Furthermore, which interpretation is needed for a particular task is not known in advance, making $\mathcal{URL}$ inherently more difficult. Considering that CUR^{2}LED aims at finding a representation effective for many tasks, CUR^{2}LED addresses *multiple interpretations of relational similarity simultaneously.* How exactly that is achieved is discussed in the next section, together with a similarity measure used for this purpose.

The final issue concerns *the structure of the feature hierarchy*. Defining such hierarchy requires the specification of the number of layers, as well as the number of hidden features (i.e., clusters) within each layer. How to automatically construct such hierarchies is currently under-explored. Consequently, the performance of these methods is sensitive to the parameter setting, requiring substantial expertise in order to choose the optimal number of features. This constitutes a major bottleneck for relational *type-aware* feature hierarchies, as separate values should be chosen for each type in the data

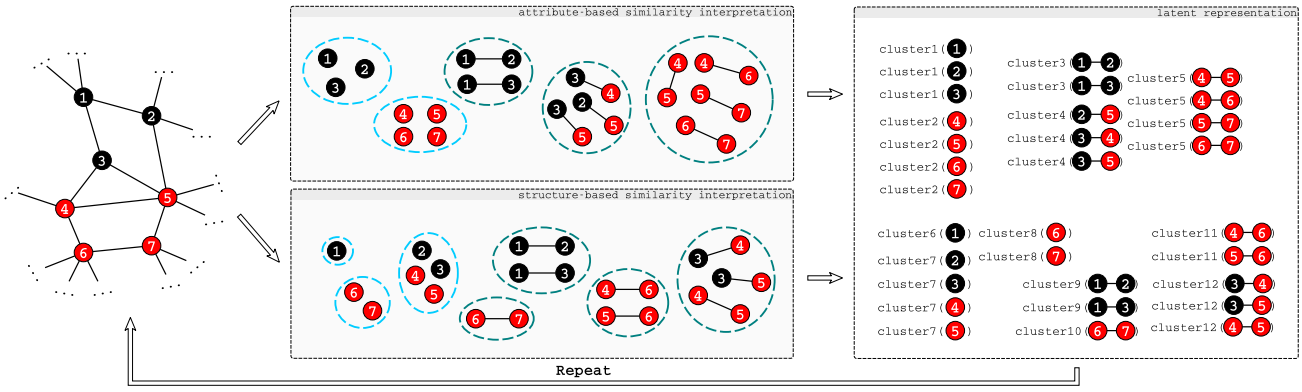


Figure 1: An illustration of CUR^{2}LED procedure. The left-most figure represents a given hypergraph, where colour of a vertex indicates its feature value. The graph (i.e., vertices and edges) is then clustered according to different similarity interpretations. The upper clustering is based on vertex attributes: the vertices are clustered into *red* and *black* ones, while the edges are clustered according to the colour of the vertices they connect. The bottom clustering is based on the structure of the neighbourhoods. The vertices are clustered into a group that have only *black* neighbours ($\{1\}$), only *red* neighbours ($\{6,7\}$), and neighbours of both colours ($\{2,3,4,5\}$). The edges are clustered into a group of edges connecting *black* vertices with only *black* neighbours and *black* vertices with *red* neighbours ($\{1\text{-}2, 1\text{-}3\}$), a group of edges connecting *red* vertices with only *red* neighbours to *red* vertices with neighbours of both colour ($\{6\text{-}7\}$), and so on. The final step transforms the obtained clusterings into a relational representation. The procedure can further be repeated to create more layers of features.

(and combination thereof for hyperedges).

To tackle this infeasibility, CUR^{2}LED builds upon a vast literature on *the problem of clustering selection* [Arbelaitz *et al.*, 2013], which is concerned with the selection of optimal number of clusters from data. Though the perfect method does not exist, automatic clustering selection mitigates the problem of the manual specification of feature hierarchies. CUR^{2}LED leverages two distinct approaches: *(1) a difference-like criterion* [Vendramin *et al.*, 2010], and *(2) a quality based criterion of Silhouette index* [Rousseeuw, 1987].

Difference-like criteria assess relative improvements on some relevant characteristic of the data (e.g. within-cluster similarity) over a set of successive data partitions produced by gradually increasing the number of clusters (N). It attempts to identify a prominent *knee* - a point when the given quality measure *saturates* and the further increase of N can offer only marginal benefit. Following the suggestion in [Vendramin *et al.*, 2010], we choose the number of clusters as the one that achieves the highest value of the following formula:

$$D(k) = \left| \frac{C(k-1) - C(k)}{C(k) - C(k+1)} \right| - \alpha \cdot k \qquad (1)$$

where $C(k)$ is the intra-cluster similarity, k is the number of clusters and α a user-specified penalty on the number of clusters.

The Silhouette index evaluates a *cohesion*, i.e., how similar an instance is to its own cluster, and a *separation*, i.e., how similar an instance is to the other clusters. It is defined as:

$$S(i) = \frac{a(i) - b(i)}{\max\{a(i), b(i)\}} \qquad (2)$$

where i is an instance, $a(i)$ is an average dissimilarity of i to the rest of the instances in the same cluster, and $b(i)$ the lowest dissimilarity of i to any other cluster. Higher values indicate a better fit of the data.

$\mathcal{C}$-representation. Once the clusters are obtained, we will represent them in the following form. For each cluster of vertices we create a unary predicate in the form of $\texttt{cluster}ID\texttt{(vertex)}$ where `vertex` is an identifier of a specific vertex. Similarly, for each cluster of hyperedges we create a n-ary predicate in the form of $\texttt{cluster}ID\texttt{(vertex}_1\texttt{,...,vertex}_n\texttt{)}$, which takes an ordered set of n vertices as arguments. We refer to the cluster-induced representation as a $\mathcal{C}$-representation.

The introduced pipeline is illustrated in Figure 1. CUR^{2}LED specified thus far describes a *meta-procedure* how to use any clustering algorithm to obtain a latent representation. In the experiments we use spectral and hierarchical clustering.

4 Similarity of relational structures

CUR^{2}LED relies on ReCeNT [Dumancic and Blockeel, 2016], a relational clustering framework focused on clustering vertices in a hypergraph. What makes ReCeNT an attractive relational clustering framework is the wide range of similarities it considers. Furthermore, which similarity is used is easily adaptable with just a few parameters. We provide a concise and intuitive description here, and refer the reader to the original paper for the details.

The core concept of ReCeNT is a *neighbourhood tree* (NT). The NT is a rooted directed graph describing a neighbourhood of a certain vertex in the hypergraph. It provides a summary of all paths that can be taken, starting from that particular vertex. The depth of a NT, i.e., how many vertices a path can contain excluding the root vertex, is pre-specified. ReCeNT compares two vertices by comparing their NTs.

This comparison is achieved by first decomposing the NT into different multisets. The multiset $V_t^l(g)$ contains all vertices of type t at distance l of a particular NT g. $E^l(g)$ is the multiset of hyperedge labels between vertices of distances l and $l+1$. Finally, $B_{t,a}^l(g)$ is the multiset of values of attribute a observed among the nodes of type t at distance l. Using only these three types of multisets, one can express a wide range of similarities. What ReCeNT considers are:

1. the similarity of the root vertices in terms of attribute values, by means of $B_{t,a}^0$
2. the similarity of attribute values of the neighbouring vertices, by means of $B_{t,a}^{l>0}$
3. the connectivity of the root vertices, by means of V_t^l
4. the similarity of neighbourhoods in terms of the vertex identities, by means of V_t^l
5. the similarity of hyperedge labels of two neighbourhoods, by means of E^l.

Each of the components represents a distinct notion of similarity. We will refer to them as *core similarities*. These core similarities can further be combined to represent more complex similarities.

4.1 Hyperedge similarity

In its original form, ReCeNT clusters vertices in a hypergraph. However, CUR^{2}LED additionally clusters hyperedges. Here we introduce a general framework for hyperedge clustering. It views hyperedges as ordered sets of vertices, and thus ordered sets of NTs.

Let $\mathcal{N}$ be a set of NTs. Let Θ denote summary operations on sets of values such as mean, minimum and maximum. Let Λ denote set operators such as union and intersection. Let $f : \mathcal{N}^2 \to \mathbb{R}$ be a similarity between two NTs, e.g. the similarity measure introduced by ReCeNT. The framework introduces two types of hyperedge similarity, namely *combination* and *merging*.

Definition 1. A *combination similarity* is a function $c : \mathcal{N}^n \times \mathcal{N}^n \times \Theta \to \mathbb{R}$ which compares two hyperedges, $e_1 = (v_1^1, ..., v_1^n)$ and $e_1 = (v_2^1, ..., v_2^n)$, by comparing the individual NTs respecting the order, $s = \big(f(v_1^1, v_2^1), \ldots, f(v_1^i, v_2^i)\big)$, and summarizing respective similarities with $\theta \in \Theta$, $\theta(s)$.

Definition 2. A *merging similarity* is a function $m : 2^{\mathcal{N}} \times 2^{\mathcal{N}} \times \Lambda \to \mathbb{R}$ which compares two hyperedges, $e_1 = (v_1^1, ..., v_1^n)$ and $e_1 = (v_2^1, ..., v_2^n)$, by first merging the NTs within a hyperedge with merging operator $\lambda \in \Lambda$, $s_1 = \lambda(v_1^1, \ldots, v_1^i)$ and $s_2 = \lambda(v_2^1, \ldots, v_2^i)$ and comparing the resulting NTs, $f(s_1, s_2)$.

Merging two NTs involves merging their respecting multisets with a merging operator λ, respecting the level. For instance, consider *set union* as λ, and g' and g'' as the NTs to be merged. Then, merging the multisets $V_t^l(g')$ and $V_t^l(g'')$ results in a multiset $V_t^l\left(\lambda(g', g'')\right) = V_t^l(g') \cup V_t^l(g'')$.

Both formulations reduce the problem to the comparison of NTs, but offer alternative views. While *merging* ignores the order of vertices in a hyperedge, *combination* respects it. Accordingly, *merging* describes the neighbourhood of a hyperedge, while *combination* examines the similarity of vertices participating in a hyperedge. In this work we use *union* as the merging operator, and *mean* as the combination operator.

4.2 Similarity interpretation

Finally, we formally introduce the notion of similarity interpretation.

Definition 3. Let $(w_1, w_2, w_3, w_4, w_5)$ be the weights associated with the *core similarities*. A *similarity interpretation* is the value assignments to the weights $(w_1, w_2, w_3, w_4, w_5)$.

Thus, it allows us to precisely control aspects of similarity considered for representation learning. For example, setting $w_1 = 1, w_{2,3,4,5} = 0$ uses only the attributes of vertices for comparison. Setting $w_3 = 1, w_{1,2,4,5} = 0$ on the other hand would identify clusters as a densely connected components. As the similarity interpretation is provided by the user, we say it *explicitly* defines the distributed representation.

5 Experiments and results

Datasets. We have used the following 6 datasets to evaluate the potential of this approach. The IMDB dataset describes a set of movies with people acting in or directing them. The UW-CSE dataset describes the interactions of employees at the University of Washington and their roles, publications and the courses they teach. The Mutagenesis dataset describes chemical compounds and atoms they consist of. The WebKB dataset consists of pages and links collected from the Cornell University's web page. The Terrorists dataset describes terrorist attacks each assigned one of 6 labels indicating the type of the attack. The Hepatitis dataset describes a set of patients with hepatitis types B and C.

Evaluation procedure. In principle, a latent representation should make learning easier by capturing complex dependencies in data more explicitly. Though that is difficult to formalize, a consequence should be that a model learned on the latent representation is (i) *less complex*, and (ii) possibly *performs better*. To verify whether that is the case with the representation created by CUR^{2}LED , we answer the following questions:

(Q1) *do representations learned by CUR^{2}LED induce models of lower complexity compared to the ones induced on the original representation?*

(Q2) *if the original data representation is sufficient to solve a task efficiently, does $\mathcal{C}$-representation preserves the relevant information?*

(Q3) *if the original data representation is not sufficient to solve the task, does a $\mathcal{C}$-representation improve the performance of a relational classifier?*

(Q4) *can the appropriate parameters for a specific dataset be found by the model selection?*

(Q5) *how does CUR^{2}LED compare to MRC, which is the closest related work?*

In order to do so, we use TILDE [Blockeel and De Raedt, 1998], a relational decision tree learner, and perform 5-fold cross validation. $\mathcal{C}$-representations and TILDE were learned on training folds, and the objects from the test fold were mapped to the $\mathcal{C}$-representation and used to test

Table 1: Performance comparison for TILDE models learned on the original and $\mathcal{C}$-representations. The first column specifies the parameters used for $\mathcal{C}$-representation, i.e., clustering algorithm (S-spectral, H-hierarchical), selection criterion and its parameter values. Both accuracies on a test set (Acc) and complexities (Cplx) are reported.

	Setup	IMDB		UWCSE		Mutagenesis		Terrorists		Hepatitis		WebKB	
		Acc	Cplx	Acc	Cplx	Acc	Cplx	Acc	Cplx	Acc	Cplx	Acc	Cplx
	Original	1.0	2.0	0.99	3.0	0.76	27.2	**0.72**	86.4	0.81	22.4	0.81	18.2
merging	S,$\alpha = 0.01$	1.0	1.0	0.99	1.2	0.79	6.6	**0.71**	34.4	0.86	19.66	**0.89**	13.6
	S,$\alpha = 0.05$	1.0	1.0	0.99	**1.0**	0.78	2.4	0.65	21.6	0.90	7.6	0.85	15.6
	S,$\alpha = 0.1$	1.0	1.0	0.99	1.2	0.78	**1.8**	0.66	32.4	0.90	6.5	**0.87**	17.8
	S,silhouette	1.0	1.0	0.99	**1.0**	0.78	**2.0**	0.6	23.6	**0.93**	**5.33**	**0.87**	14.8
	H,$\alpha = 0.01$	1.0	1.0	0.98	4.4	**0.83**	**2.0**	0.48	**9.4**	0.86	12.0	0.83	12.6
	H,$\alpha = 0.05$	1.0	1.0	0.99	4.2	**0.83**	**2.0**	0.48	11.6	0.82	16.0	0.69	27.2
	H,$\alpha = 0.1$	1.0	1.0	0.99	4.0	0.79	5.2	0.47	**8.8**	0.82	13.4	0.61	32.2
	H,silhouette	1.0	1.0	0.98	**1.0**	0.80	3.4	0.47	13.0	**0.93**	8.66	0.68	18.0
combination	S,$\alpha = 0.01$	1.0	1.0	0.99	1.2	0.79	**2.0**	**0.72**	24.0	0.90	7.6	**0.91**	11.8
	S,$\alpha = 0.05$	1.0	1.0	0.99	**1.0**	0.79	**2.0**	0.69	22.8	0.88	12.2	0.86	**10.0**
	S,$\alpha = 0.1$	1.0	1.0	1.0	**1.0**	0.76	**2.0**	0.66	16.8	0.90	12.6	0.87	17.0
	S,silhouette	1.0	1.0	0.99	**1.0**	0.77	**2.0**	0.6	24.2	**0.93**	16.4	**0.88**	13.8
	H,$\alpha = 0.01$	1.0	1.0	0.99	2.8	0.79	4.0	0.51	30.6	0.80	29.33	0.83	12.6
	H,$\alpha = 0.05$	1.0	1.0	0.99	2.8	0.78	2.8	0.51	30.6	0.82	16.33	0.69	27.2
	H,$\alpha = 0.1$	1.0	1.0	0.99	2.8	0.78	11.0	0.50	27.3	0.78	14.0	0.61	32.2
	H,silhouette	1.0	1.0	0.99	2.0	0.80	4.0	0.50	30.0	0.83	11.6	0.68	18.0
MRC	$\lambda = -1$	1.0	1.0	0.93	21.0	0.6	0	0.64	138.7	0.61	99.4	0.64	44.4
	$\lambda = -5$	1.0	1.0	0.95	25.9	0.63	23.5	0.50	126.5	0.84	64.8	0.68	40.0
	$\lambda = -10$	1.0	1.0	0.96	13.7	0.72	35.0	0.51	102.1	0.57	5.7	0.66	40.8

TILDE. The following similarity interpretation were used for each dataset: (0.5,0.5,0.0,0.0,0.0), (0.0,0.0,0.33,0.33,0.34), (0.2,0.2,0.2,0.2,0.2). The first set of weights uses only the attribute information, the second one only the link information, while the last one combines every component.

As a complexity measure of a model we use the number of nodes a trained TILDE model has. We use the following values for the α parameter in Equation 1: $\{0.1, 0.05, 0.01\}$. In the case of MRC, we used the following values for the λ parameter: $\{-1, -5, -10\}$. The λ parameter has the same role as α in the proposed approach, affecting the number of clusters chosen for each type[2].

Results

To answer the above mentioned questions, we perform two types of experiments. Table 1 summarizes the results of cross validation. The accuracies on test set and the complexities of TILDE models are stated for both original and $\mathcal{C}$-representations. Table 2 summarizes the results of the model selection where we dedicate one fold as a *validation set*, and perform the cross validation on the remaining folds to identify the best parameter values (i.e., the choice of a clustering algorithm, a clustering selection procedure and the appropriate hyperedge similarity) for each dataset.

Q1. Table 1 shows that the models learned on $\mathfrak{C}$-representation consistently have lower complexity than the ones learned on the original data. That is especially the case when $\mathcal{C}$-representation is obtained by spectral clustering, which consistently results in a model of a lower complexity. The reduction of complexity can even be surprisingly substantial, for instance on the Mutagenesis and Hepatitis datasets where the model complexities are reduced by factors of 10 and 4, respectively. When the $\mathcal{C}$-representation is obtained with hierarchical clustering, models of lower complexity are obtained on all datasets except the WebKB and UWCSE datasets. These results suggest that the $\mathcal{C}$-representation in general makes complex dependencies easier to detect and express.

Q2. The IMDB and UWCSE datasets are considered as *easy* relational datasets, where the classes are separable by a single attribute or a relationship. Thus, TILDE is able to achieve almost perfect performance with the original data. The original representation is therefore sufficient to solve the task, and we are interested whether the relevant information will be preserved within the $\mathcal{C}$-representation. The results in Table 1 do suggest so, as TILDE achieves identical performance regardless of the representation.

Q3. The remaining datasets are more difficult than the previously discussed ones. On the Mutagenesis, Hepatitis and WebKB datasets, $\mathcal{C}$-representation improves the performance. On the Terrorists dataset, however, no improvement in performance is observed. What distinguishes this dataset from the others is that it contains only two edge types (indicating co-located attack, or ones organized by the same organization), an abundant number of features, while other datasets are substantially more interconnected. Thus, focusing on the relational

[2]Note that it is difficult to exactly match the values of α and λ as both methods operate on different scales, and the authors do not provide a way how to choose an appropriate value

Table 2: Model selection results. For each dataset, a selected parameters are reported together with the accuracies on the training and test sets. The first element indicates the selected clustering algorithm (S-spectral, H-hierarchical), the second one the clustering selection criteria, while the last one indicates the hyperedge similarity (C-combination, M-merging). The last column indicates the performance on the original data representation.

Dataset	Parameters	Training	Validation	Original
IMDB	all	1.0	**1.0**	**1.0**
UWCSE	S, silhouette, C	0.99	**1.0**	0.99
Mutagenesis	H, α=0.01, M	0.86	**0.84**	0.79
Hepatitis	S, silhouette, M	0.92	**0.89**	0.8
WebKB	S, α=0.01, C	0.88	**0.88**	0.79
Terrorists	S,α=0.01, C	0.70	0.69	**0.71**

information is not as beneficial as the features themselves.

These results suggest that $\mathcal{C}$-representations indeed improve performance of the classifier, compared to the one learned on the original data representation. First, the $\mathcal{C}$-representation created with spectral clustering consistently performs better on all datasets, except the Terrorists one. Second, if the learning task does not have a strong relational component, then $\mathcal{C}$-representations are not beneficial and can even hurt the performance. Third, the choice of a clustering algorithm matters, and spectral clustering does a better job in our experiments - it always results in improved or at least equally good performance. Fourth, the choice of treating hyperedges as ordered (by *combination*) or unordered (*merging*) sets is data-dependent, and the difference in performance is observed.

Combining the results from **Q1**, **Q2** and **Q3** shows that *the main benefit* of CUR^{2}LED is *the transformation of data such that it becomes easier to express complex dependencies*. Consequently, the obtained models have lower complexities and their performance often improves.

Q4. To ensure that the previously discussed results do not over-fit the data, we additionally perform model selection. We dedicate one fold as the *validation set*, and use the remaining folds to find the best parameter values of both CUR^{2}LED and TILDE. Table 2 summarizes the results and reports the selected choice of parameter, together with the performance on the validation set. These results are consistent with the ones in Table 1: $\mathcal{C}$-representation improves the performance in the majority of cases, and the selected parameters correspond to the best performing ones in Table 1.

Considering the computational cost, CUR^{2}LED is an expensive step which depends on the size of a dataset like all representation learning approaches. Performing a 5-fold cross validation on a single CPU took approximately 5 minutes for the IMDB and UWCSE datasets, 24 hours for the Terrorists dataset and approximately a week for the remaining datasets. Though expensive, latent representation has to be created only once and can be reused for many tasks with the same dataset. Moreover, CUR^{2}LED is easily parallelizable which can substantially improve its efficiency.

Q5. Table 1 shows that CUR^{2}LED substantially outperforms MRC on all datasets, achieving better performance on all datasets except the IMDB. Moreover, MRC rarely shows benefit over the original data representation, with an exception on the Hepatitis dataset. Considering the model complexity, the models learned on MRC-induced representation are substantially more complex than the ones learned on $\mathcal{C}$-representations. Table 3 summarize the number of clusters created by CUR^{2}LED and MRC. One can see that MRC creates substantially more clusters than CUR^{2}LED . Because of this, the found clusters contain only a few objects which makes it difficult to generalize well, and increases the model complexity. The number of clusters found by CUR^{2}LED is relatively high, because it finds a representation of data suitable for many classification tasks over the same datasets. Thus, most of the features are redundant for one specific task, but clearly contain better information as the models learned on them perform better and have lower complexity.

Table 3: Vocabulary sizes. M indicates MRC, while S and H indicate CUR^{2}LED representations with spectral and hierarchical clustering, respectively. Vocabulary sizes obtained with *merging* and *combination* similarities were similar, so only the one for merging is reported.

Setup	UW	Muta	WebKB	Terror	IMDB	Hepa
Original	10	12	775	107	5	22
S, $\alpha = 0.01$	109	53	65	30	75	85
S, $\alpha = 0.05$	87	37	63	26	69	66
S, $\alpha = 0.1$	72	31	57	24	59	28
S, silhouette	93	17	59	37	74	79
H, α=0.01	93	38	64	25	69	62
H, α=0.05	85	34	64	20	65	50
H, $\alpha = 0.1$	68	22	58	18	55	46
H, silhouette	85	20	55	43	64	61
M, λ=-1	183	535	817	318	49	655
M, λ=-5	140	346	331	116	38	297
M, λ=-10	49	224	219	91	18	120

6 Conclusion

This work introduces CUR^{2}LED - a clustering-based framework for unsupervised representation learning with relational data, which describes both instances and relationships between them. Viewing relational data as hypergraph, CUR^{2}LED learns new features by clustering both instances and their relationships. i.e., vertices and hyperedges in the corresponding hypergraph. To support such procedure, we introduce a general hyperedges clustering framework based on similarity of vertices participating in the hyperedge. A distinct feature of CUR^{2}LED is the way it uses the ambiguity of similarity within relational data, i.e., whether two relational objects are similar due to their features of relationships, to generate distributed representation of data. We design several experiments to verify the usefulness of latent representation generated by CUR^{2}LED . The results show that the latent representations created by CUR^{2}LED provide a better representation of data that results in models of lower complexity and better performance. In future work, we will extend CUR^{2}LED towards semi-supervised settings, and investigate alternative ways for learning a distributed representations directly from data.

References

[Arbelaitz *et al.*, 2013] Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, JesúS M. PéRez, and IñIgo Perona. An extensive comparative study of cluster validity indices. *Pattern Recogn.*, 46(1):243–256, January 2013.

[Bengio *et al.*, 2007] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *NIPS*. MIT Press, 2007.

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, August 2013.

[Bengio, 2009] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[Blockeel and De Raedt, 1998] Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(12):285 – 297, 1998.

[Bordes *et al.*, 2011] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 301–306. AAAI Press, 2011.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 2787–2795, USA, 2013. Curran Associates Inc.

[Coates *et al.*, 2011] A. Coates, H. Lee, and A.Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 215–223. JMLR W&CP, 2011.

[Craven and Slattery, 2001] Mark Craven and Seán Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Mach. Learn.*, 43(1-2):97–119, April 2001.

[Dumancic and Blockeel, 2016] S. Dumancic and H. Blockeel. An efficient and expressive similarity measure for relational clustering using neighbourhood trees. *ArXiv e-prints, 1604.08934*, April 2016.

[Henaff *et al.*, 2015] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.

[Hinton and Salakhutdinov, 2006] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[Kok and Domingos, 2007] Stanley Kok and Pedro Domingos. Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 433–440, New York, NY, USA, 2007. ACM.

[Kok and Domingos, 2008] Stanley Kok and Pedro Domingos. Extracting semantic networks from text via relational clustering. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pages 624–639, Berlin, Heidelberg, 2008. Springer-Verlag.

[Kramer, 1995] Stefan Kramer. Predicate Invention: A Comprehensive View. *Technical report*, 1995.

[Niepert *et al.*, 2016] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2014–2023, 2016.

[Niepert, 2016] Mathias Niepert. Discriminative gaifman models. In *Advances in Neural Information Processing Systems 29*, pages 3405–3413. Curran Associates, Inc., 2016.

[Popescul and Ungar, 2004] Alexandrin Popescul and Lyle H. Ungar. Cluster-based concept invention for statistical relational learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 665–670, New York, NY, USA, 2004. ACM.

[Ranzato *et al.*, 2007] Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *NIPS*, pages 1185–1192. Curran Associates, Inc., 2007.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, February 2006.

[Rousseeuw, 1987] Peter Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, November 1987.

[Vendramin *et al.*, 2010] Lucas Vendramin, Ricardo J. G. B. Campello, and Eduardo R. Hruschka. Relative clustering validity criteria: A comparative overview. *Stat. Anal. Data Min.*, 3(4):209–235, August 2010.

[Yang *et al.*, 2014] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575, 2014.

Demystifying Relational Latent Representations

Sebastijan Dumančić and Hendrik Blockeel

Department of Computer Science, KU Leuven Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract. Latent features learned by deep learning approaches have proven to be a powerful tool for machine learning. They serve as a data abstraction that makes learning easier by capturing regularities in data explicitly. Their benefits motivated their adaptation to the relational learning context. In our previous work, we introduce an approach that learns *relational* latent features by means of clustering instances and their relations. The major drawback of latent representations is that they are often *black-box* and difficult to interpret. This work addresses these issues and shows that (1) latent features created by clustering are interpretable and capture interesting properties of data; (2) they identify local regions of instances that match well with the label, which partially explains their benefit; and (3) although the number of latent features generated by this approach is large, often many of them are highly redundant and can be removed without hurting performance much.

Keywords: relational learning, deep learning, unsupervised representation learning, clustering

1 Introduction

Latent representations created by deep learning approaches [1] have proven to be a powerful tool in machine learning. Traditional machine learning algorithms learn a function that directly maps data to the target concept. In contrast, deep learning creates several layers of latent features between the original data and the target concept. This results in a multi-step procedure that simplifies a given task before solving it.

The progress in learning such latent representations has predominantly focused on vectorized data representations. Likewise, their utility has been recognized in the relational learning community [2] in which models are learned not only from instances but from their relationships as well [3,4]. There the problem is known as *predicate invention* [5,6]. The prevalent latent representations paradigm in that direction are *embeddings to vector spaces* [7,8,9]. The core idea behind the embeddings is to replace symbols with numbers and logical reasoning with algebra. More precisely, relational entities are transformed to low-dimensional vectors and relations to vectors or matrices. This way of learning latent features corresponds to learning the low-dimensional representations of relational entities and relations. Many variations of this formalization exist, but

they share the same underlying principles. Assuming facts `p(a,b)` and `r(b,a)`, `a` and `b` are entities whereas `p` and `r` are existing relations between them. One major line of research interprets relations as *translations between facts in Euclidean space*. More precisely, given the fact `p(a,b)` the goal is to find vector representations of `a`, `b` and `p` (namely a, b and p) such that $a + p \approx b$ and $b + r \approx a$. The other major line of research interprets embeddings as a *factorization of a knowledge base*. These approaches represent entities as vectors and relations as matrices; the goal is to find vector representations of entities and relations (vectors a and b for `a` and `b`, matrices P and R for `p` and `R`) such that products aPb and bRa have *high values*. In contrast, given a false statement `q(a,b)` product aQb should have a *low value*.

These embeddings approaches have several drawbacks. First, the latent features created that way have no inherent meaning – they are created to satisfy the aforementioned criteria. This is thus a major obstacle for interpretability of the approach, which is important in many aspects and one of the strengths of relational learning. Second, huge amounts of data are needed in order to extract useful latent features. Knowledge bases used for training often contain millions of facts. Third, it is not clear how these approaches can handle unseen entities (i.e., an entity not present in the training set and whose embedding is therefore not known) without re-training the entire model. Fourth, compared to the full-fledged expressivity of statistical relational learning [3] these approaches have reduced expressivity.

Recently, Dumančić and Blockeel [10] introduced a complementary approach, titled CUR^{2}LED, that takes a relational learning stance and focuses on learning relational latent representations in an unsupervised manner. Viewing relational data as a hypergraph in which instances form vertices and relationships among them form hyperedges, the authors rely on clustering to obtain latent features. The core component in this approach is a declarative and intuitive specification of the similarity measure used to cluster both instances and their relationships. This consequently makes entire approach more *transparent* with respect to the meaning of latent features, as the intuitive meaning of similarity is precisely specified.

The benefits of latent representations were clearly shown with respect to both performance and complexity. The complexity of models learned on latent features was consistently lower compared to the models learned on the original data representation. Moreover, the models learned with latent features often resulted in improved performance, by a large margin as well. These two results jointly show that latent representations capture more complex dependencies in a simple manner.

In this work we further investigate the properties of relational latent representations created by CUR^{2}LED. We start by asking the question: *what do latent features mean?* We introduce a simple method to extract the meaning of the latent features, and show that they capture interesting properties. We ask next: *what makes latent representations effective?* The initial work showed the benefits of the latent representations, however, no explanation is offered why

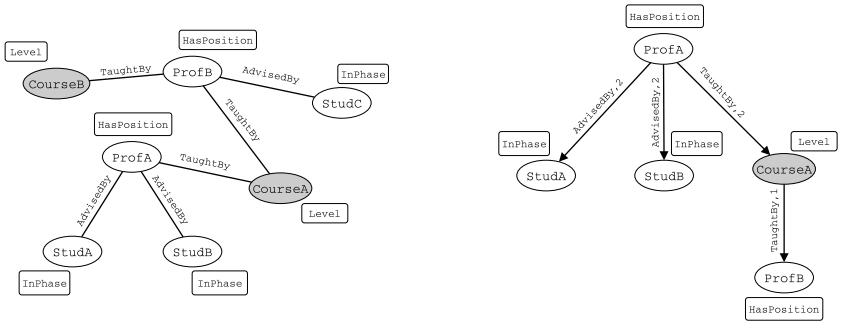


Fig. 1. A snapshot of a knowledge base (left) and the corresponding neighbourhood trees of `ProfA` entity (right). The knowledge base describes students, professors and courses they teach. Entities (people and courses) are represented with node, their attributes with rectangles and relationships with edges. Attribute values are left out for brevity.

that is the case. We hope to shed light behind the scene and offer (at least a partial) answer why that is the case.

In the following section we first briefly introduce neighbourhood trees – a central concept of CUR^{2}LED. We then describe an approach used in extracting the knowledge form the latent features, and investigating the properties of such latent representation. The results are presented and discussed next, followed by the conclusion.

2 Background

2.1 Neighbourhood trees

The central concept of CUR^{2}LED is a neighbourhood tree [11]. The neighbourhood tree is a rooted directed graph describing an instance, together with instances it relates to and their properties. Viewing relational data as a hypergraph, the neighbourhood tree provides a summary of all path of a pre-defined length that originate in a particular vertex (see Figure 1).

As instances are represented as neighbourhood trees, two instances are compared by comparing corresponding neighbourhood trees. The authors introduce a versatile and declarative similarity measure [11] that analyses neighbourhood trees over multiple aspects by introducing the following *core similarities*:

- attribute similarity of root vertices
- attribute similarity of neighbouring vertices
- connectivity between root vertices
- similarity of vertex identities in a neighbourhood
- similarity of edge types

Continuing the example in Figure 1, person instances can be clustered based on their own attributes, which yields clusters of professors and students. Clustering person instances based on the vertex identities in their neighbourhood yields clusters of *research groups* – a professor and his students.

In order to obtain the core similarities, a neighbourhood tree is first decomposed into three multisets:

- the multiset of vertices of type t at depth l, with the root having depth 0 (V_t^l)
- the multiset of values of attribute a observed among the nodes of type t at depth l ($B_{t,a}^l$)
- the multiset of edge types between depth l and $l+1$ (E_l).

Each of the core similarities is now defined in terms of comparing relative frequencies of elements in the aforementioned multisets. For instance, the attributes similarity of root vertices is achieved by comparing relative frequencies of elements in $B_{t,a}^0$ (for each attribute a), while the attributes similarity of neighbouring vertices is achieved by comparing $B_{t,a}^{>1}$ (for each vertex type t and attribute a). Similarly, the similarity of edge types is achieved by comparing relative frequencies of elements in E_l for each level l of a neighbourhood tree. This comparison of relative frequencies of elements in a neighbourhood tree is the central notion we will exploit when discovering the meaning of latent features.

These core similarities form basic building blocks for a variety of similarity measure, all defined over neighbourhood trees. The final similarity measure is a linear weighted combination of the core similarities. Weights simply define a relative importance of core similarities in the final similarity measure. The value assignments to the weights defines a *similarity interpretation*. For more details of the approach we refer the reader to [11].

2.2 CUR^{2}LED

Two ideas are central to CUR^{2}LED. First, it learns latent features by clustering instances and their relationships. In order to cluster relationships, it makes a straightforward interpretation of relationships as sets of vertices (ordered or not). Second, it uses multiple similarity interpretations (i.e., combinations of core similarities) to obtain a variety of features. This is inspired by the notion of *distributed representation*, one of the pillars of representation learning. Distributed representation refers to a notion of a *reasonable sized representation capturing a hug number of possible configurations* [12]. A common view is that a distributed representation represents concepts with independently manipulated factors, instead of a single one with one-hot representations. Both ideas are realised by means of neighbourhood trees. Instances and relationships are represented as (collections of) neighbourhood trees, while using different similarity interpretations (which consider only certain parts of neighbourhood trees) explicitly defines a distributed representation.

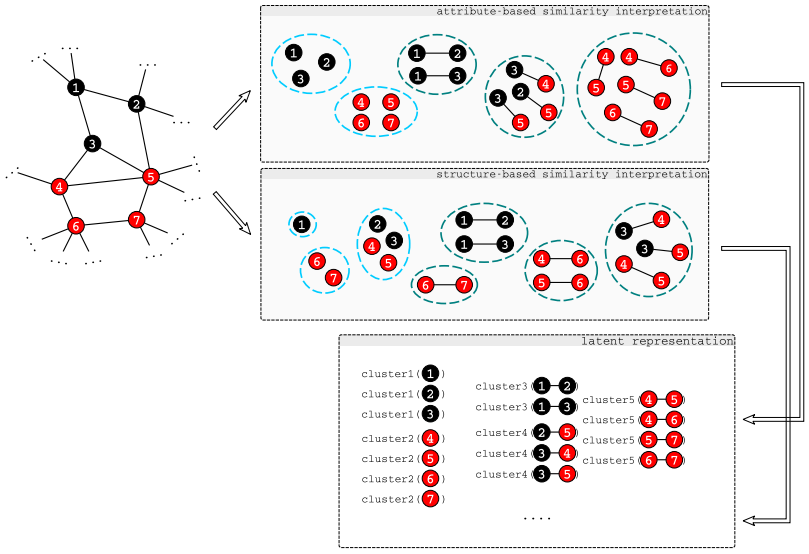


Fig. 2. (Figure and example taken from [10]) Starting from a relational data see as a hypergraph, CUR^{2}LED clusters the vertices and hyperedges in the hypergraph according to different similarity interpretations. It first performs clustering based on the vertex attributes (indicated by the colour of vertices): the vertices are clustered into *red* and *black* ones, while the edges are clustered according to the colour of the vertices they connect. It then performs clustering based on the structure of the neighbourhoods (the bottom part). The vertices are clustered into clusters that have (i) only *black* neighbours ({1}), (ii) only *red* neighbours ({6,7}), and (iii) neighbours of both colours ({2,3,4,5}). The edges are clustered into clusters of (i) edges connecting *black* vertices with only *black* neighbours and *black* vertices with *red* neighbours ({1-2,1-3}), (ii) edges connecting *red* vertices with only *red* neighbours to *red* vertices with neighbours of both colour ({6-7}), and so on. The final step represents the obtained clusterings in the format of first-order logic.

Latent features are thus learned through *repeated clustering of instances and relations and alternating the similarity measure in each iteration* (see Figure 2). Each latent feature, corresponding to a cluster of instances, is associated with one latent predicate. Truth instantiations of latent predicates reflect the cluster assignments, i.e., the instantiations of a latent predicate are true for instances that belong to the cluster; therefore, latent features are defined extensionally and lack an immediate interpretable definition.

3 Opening the black box of latent features

The primary focus of this work is on understanding *what* do latent features created by CUR^{2}LED mean and *why* do they prove useful. We start by describing the procedure to extract the meaning of the latent features.

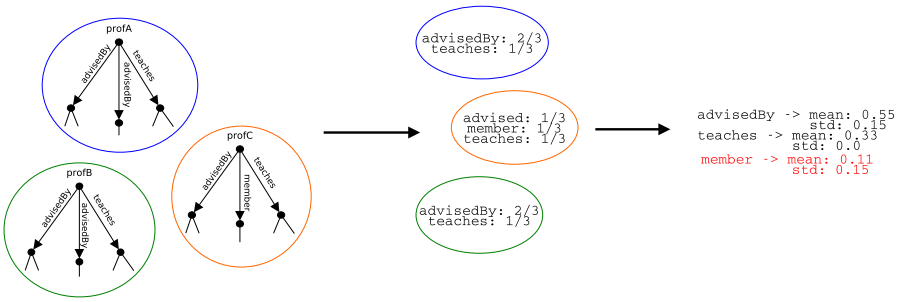


Fig. 3. Discovering the meaning of latent features by analysing their relations. Properties that describe latent features are the ones that have similar relative frequency in all neighbourhood trees. Starting from a cluster of instances viewed as neighbourhood trees (left), the relative frequencies of elements are calculated for each neighbourhood tree (middle). Next, the mean and standard deviation of relative frequencies are calculated for each individual element within the cluster (right). Which elements *explain* the latent features is decided with θ-confidence. Setting θ to 0.3 identifies `advisedBy` and `teaches` as relevant elements (in black).

3.1 Extracting the meaning of latent features

Although the latent features discovered by CUR^{2}LED are defined extensionally, the intuitive specification of the similarity measure (and its core similarities) makes CUR^{2}LED a transparent method with a clear description which elements of neighbourhood trees make two instances similar. Consequently, discovering the meaning of latent features is substantially easier than with the embedding approaches (and deep learning in general).

Each latent feature corresponds to a cluster and the meaning of the features is reflected in the *prototype* of the cluster. To approximate the *mean* or *prototypical* neighbourhood tree, we search for the elements common to all neighbourhood trees forming a cluster. These elements can be either attribute values, edge types or vertex identities. The similarity interpretations used to obtain the cluster limits which elements are considered to be a part of a definition. Moreover, neighbourhood trees [11] are compared by the relative frequencies of their elements, not the existence only. Therefore, to find a mean neighbourhood tree and the meaning of a latent feature, we search for *the elements with similar relative frequencies within each neighbourhood tree forming a cluster*.

To identify such elements, we proceed in three steps illustrated in Figure 3.

1. **Calculate the relative frequencies of all elements within each individual neighbourhood tree, per level and vertex type**. In case of discrete attributes, that corresponds to a distribution of its values. In case of numerical attributes, we consider its mean value. In case of vertex identities

and edge types, we simply look into their frequencies with respect to the depth in a neighbourhood tree. In the example in Figure 3, the neighbourhood tree for `profA` contains two `advisedBy` relations, thus its frequency is $\frac{2}{3}$.

2. **Calculate the mean and standard deviation of relative frequency for each element within a cluster**. In Figure 3, the frequencies of the `advisedBy` elements in individual neighbourhood trees are $\frac{2}{3}$, $\frac{2}{3}$and$\frac{1}{3}$. Thus, its mean is 0.55 with a standard deviation of 0.15.
3. **Select relevant elements.** The final step involves a decision which elements should form a definition of a latent feature. Relevant elements are identified by a notion of θ-*confidence* which captures the allowed amount of variance in order to element to be relevant.

Definition 1. ***(θ-confidence)*** *An element with mean value μ and standard deviation σ in a cluster, is said to be θ-confident if $\sigma \in [0, \theta \cdot \mu]$.*

In Figure 3, setting θ to 0.3 makes `advisedBy` a 0.3-confident element, because its standard deviation of 0.15 is within the range $[0, 0.3 \cdot 0.55] = [0, 0.165]$ specified by θ. In contrast, `member` is not a 0.3-confident elements as its standard deviation is outside the range $[0, 0.3 \cdot 0.11] = [0, 0.0363]$.

The above-described procedure explains the latent features in terms of distribution of the elements in the neighbourhood of an instance, which has its pros and cons. On the downside, this type of explanation does not conform to the standard first-order logic syntax common within relational learning. Despite this reduced readability, these explanations are substantially more transparent and interpretable than the ones produced by the embeddings approaches. However, one benefit of this approach is that it increases the expressivity of a relational learner by extensionally defining properties otherwise inexpressible in the first-order logic.

3.2 Properties of latent spaces

Latent features produced by CUR^{2}LED have proven useful in reducing the complexity of models and improving their performance. However, no explanation was offered why that is the case. In the second part of this work, we look into the properties of these latent representations and offer a partial explanation for their usefulness. To answer this question we introduce the following properties: label entropy, sparsity and redundancy.

Entropy and sparsity. Label entropy and sparsity serve as a proxy to a quantification of learning difficulty – i.e., how difficult is it to learn a definition of the target concept. Considering a particular predicate, label entropy reflects a *purity* of its true groundings with respect to the provided labels. Intuitively, if true groundings of predicates tend to predominantly focus on one particular label, we expect model learning to be easier.

Sparse representations, one of the cornerstones of deep learning [12], refer to a notion in which concepts are explained based on local (instead of global)

properties of instance space. Even though many properties might exist for a particular problem, sparse representations describe instances using only a small subset of those properties. Intuitively, a concept spread across a small number of local regions is expected to be easier to capture than a concept spread globally over an entire instance space.

Quantifying sparsity in relational data is a challenging task which can be approached from multiple directions – either by analysing the number of true groundings or interaction between entities, for instance. We adopt a simple definition: the number of true groundings of a predicate.

Label entropy and sparsity jointly describe a compelling property of data representation – instances space is divided in many local regions that match labels well and consequently make learning substantially easier.

Redundancy. A downside of CUR^{2}LED is the high number of created features. Despite their proven usefulness, a high number of latent features enlarges the search space of a relational model and increases the difficulty of learning. As similarity interpretations are provided by the user, it is possible that almost identical clusterings are obtained with different similarity interpretations. Thus, if many of the features are redundant, removing them simplifies learning.

We measure the redundancy with the *adjusted Rand index* (ARI) [13], a standard measure for overlap between clusterings, and study its impact on the performance. To evaluate the influence of redundant features, we modify CUR^{2}LED by adding an additional *overlap parameter* α. Every time a new clustering is obtained, we check its overlap with the previously discovered clusterings using the ARI. If the calculated value is bigger than α, the clustering is rejected.

4 Experiments and results

We devise the experiments to answer the following questions:

(Q1) *Are latent features created by CUR^{2}LED interpretable and do they capture sensible information?*

(Q2) *Do latent features that result in models of lower complexity and/or improved performance exhibit a lower label entropy compared to the original data representation?*

(Q3) *Are latent representation that improve the performance of a model sparser than the original data representations?*

(Q4) *To which extent are latent features redundant?*

4.1 Datasets and setup

The results obtained in [10] can be divided in three categories. The first category contains the IMDB and UWCSE datasets; these datasets present easy relational learning tasks in which the original data representation is sufficient for almost perfect performance. The main benefit of latent representations for these tasks was the reduction of model complexity. The second category includes the TerroristAttack dataset, in which the main benefit of latent representation was the

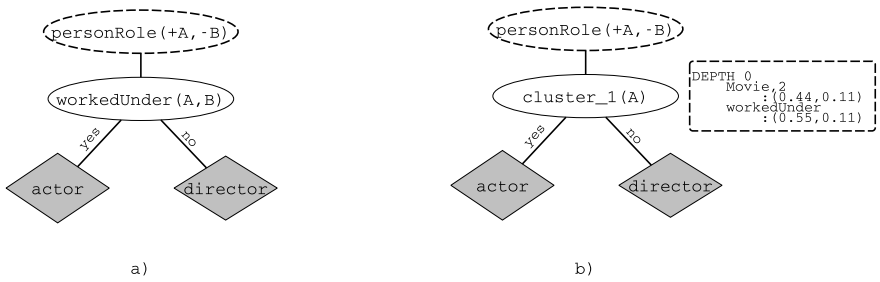


Fig. 4. Relational decision trees learned on the original (left) and latent (right) data representation of the IMDB dataset. The dashed ellipse indicates the target predicate and its arguments. The first argument, marked `A` and declared as input (`+`), denotes a person. The second argument, marked `B` and declared as output (`-`), states the label of the instance given by `A`. The values in the leaves of the decision trees are assignments to `B`. The dashed rectangle describes the latent feature – for each level of the *mean neighbourhood tree*, θ-confident elements are listed with the mean and standard deviation.

reduction of complexity, but not the performance. The third category involves the Hepatitis, Mutagenesis and WebKB datasets. These tasks benefited from latent representations in both performance and reduced model complexity. That is especially true for the Hepatitis and WebKB datasets on which the performance was improved by a large margin.

We take a representative task from each of the categories. Precisely, we use IMDB, UWCSE, Hepatitis and TerroristAttack datasets in our experiments. Both IMDB and UWCSE datasets were included as they are easy to understand without the domain knowledge, and thus useful for analysing the interpretability of relational latent features. As for the parameters of latent representation, we take the best parameters on individual datasets selected by the model selection procedure in [10]. When analysing the interpretability, we set θ to 0.3.

When evaluating the redundancy, we create latent representations by setting the α to the following values: $\{0.9, 0.8, 0.7, 0.6, 0.5\}$. We then learn a relational decision tree TILDE [14] on the obtained representation and compare accuracies, the number of created features and the number of facts.

When analysing the entropy and sparsity of representations, predicates indicating labels (such as `Professor` or `Student`) and entity definitions (such as `Person` or `Course`) are not considered in the analysis.

4.2 Interpretability

To illustrate the interpretability of relational features, we show examples of latent features created for two easy to interpret dataset - IMDB and UWCSE. We show that the relational decision trees learned on both original and latent representations. The explanations of latent features are provided as well.

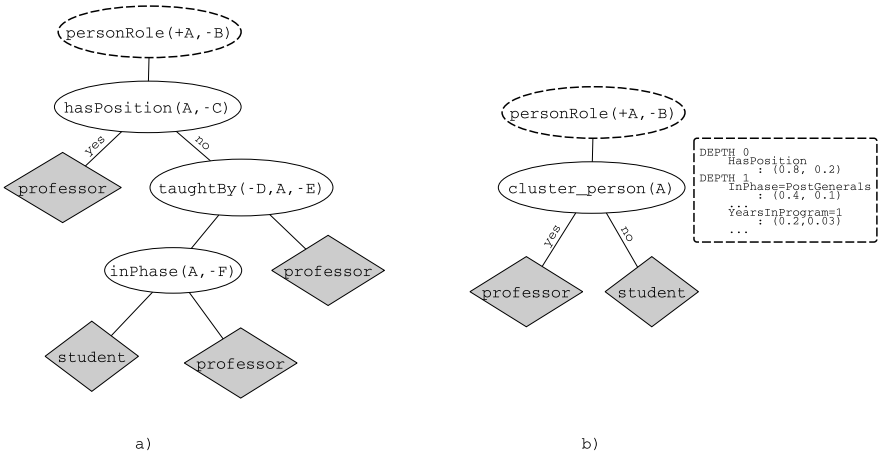


Fig. 5. Relational decision trees learned on the original (left) and latent (right) representations of the UWCSE dataset. The elements have the same meanings as in Figure 4.

Figure 4 shows the decision trees learned on the IMDB dataset. The task is to distinguish between actors and directors – this is a simple relational learning task and both original and latent decision tree achieve the perfect performance with only a single node. Even though latent representation does not seem beneficial in this particular case, it is interesting to see that the selected latent feature captures the same information as the decision tree learned on the original data – person instances in `cluster_1` are the ones that have a relationship with `movie` instances, and have worked under another person (a director).

Figure 5 shows the decision trees for the UWCSE dataset, which benefit from the latent features. Despite the simplicity of distinguishing students from professors, the decision tree learned on the latent features is more compact and has only a single node whereas the decision tree learned on the original features consists of three nodes. The latent feature here again captures similar knowledge as the original decision tree but expressed in a simpler manner – professor is someone who either has a position at the faculty, or is connected to people who are currently in a certain phase of a study program and have been in the program for a certain number of years.

What is particularly interesting about the examples above is that, even though the latent features are created in an unsupervised manner, they match the provided label very well. Moreover, they seem to almost perfectly capture the labelled information as only a few features are needed to outperform the decision tree learned on the original data representation. This observation shows that CUR^{2}LED is indeed capturing sensible knowledge in the latent space.

Both aforementioned examples are easy to understand and interpret without an extensive domain knowledge. The other tasks that have benefited more from the latent features are substantially more difficult to understand. For instance,

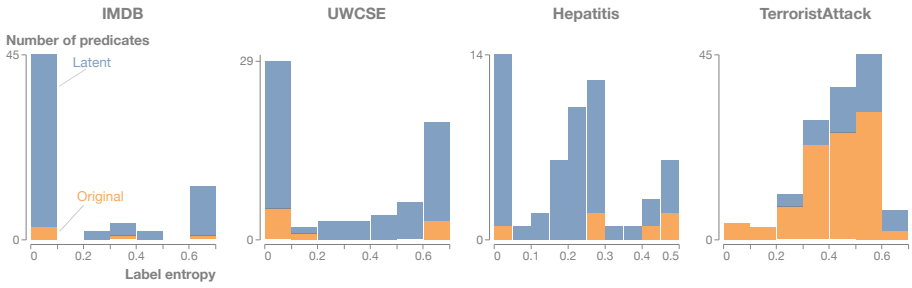


Fig. 6. Latent representations for IMDB, UWCSE and Hepatitis datasets contain substantially larger number of predicates (and the corresponding facts) with low label entropy, compared to the original representation of data. On the TerroristAttack dataset, for which the latent representation has not been useful, that is not the case - both original and latent representation demonstrate similar trends in label entropy of the predicates and the corresponding facts.

the latent features created from the Mutagenesis dataset reduce the complexity of the relational decision tree from 27 to only 3 nodes, while improving the accuracy for 4 %. Similarly, on the Hepatitis dataset the latent features reduced the complexity of a decision tree from 22 nodes down to 5, improving the accuracy for 11 %. Because these examples require an extensive knowledge to interpret them, we leave them out from this work.

4.3 Properties of latent spaces

Label entropy. Figure 6 summarizes the label entropy for each dataset. In all cases where representation learning proved helpful (i.e., IMDB, UWCSE, Hepatitis), latent representations have a substantially larger number of predicates with low label entropy compared to the original data representation. The latent representation for the TerroristAttack datasets, however, shows a different behaviour in which latent features with high entropy dominate the representation. These results agree with the expectation that a high number of low entropy features makes learning easier. However, not all latent features have low label entropy. This is expected, as the labels are not considered during learning of latent features. It also does not pose a problem – these latent features are less consistent with the one particular task, but it might well be the case that those features are useful for a different task.

Sparsity. Figure 7 summarizes the sparsity results in terms of the number of true instantiations of predicates. The distribution of the number of true groundings in the latent representations (where latent features are beneficial) is heavily skewed towards a small number of groundings, in contrast with the original representation. That is especially the case with the Hepatitis dataset, which profits the most from the latent features. The exception to this behaviour is again the TerroristAttack dataset in which the original representation already

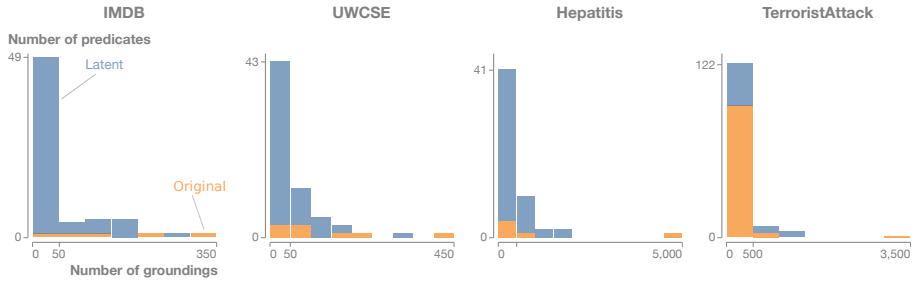


Fig. 7. Latent representation tends to be sparser than the original representation on the datasets where it is beneficial (IMDB, UWCSE and Hepatitis). On the TerroristAttack dataset, where the latent representation is not beneficial, both original and latent representation follow the same trend.

is very sparse. These results indicates that latent features indeed describe smaller groups of instances and their local properties, instead of global properties of all instances.

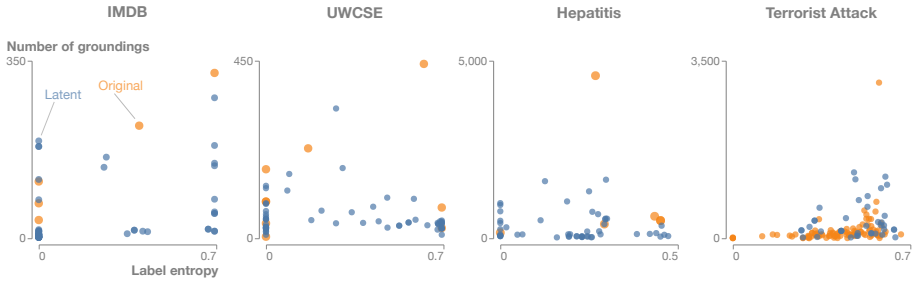


Fig. 8. Contrasting the label entropy of predicates and the number of true groundings reveals that the many latent predicates with the low label entropy have similar number of groundings as the predicates of the original data representation. This means that the trivial case, in which a large number of low-entropy predicates is obtained due to many predicates that have just a few true groundings, is not explanation for the experimental results. Instead, the latent representation, when beneficial, successfully identifies local regions in the instance space that match well with the provided labels. The exception to this is again the TerroristAttack dataset.

Connecting label entropy and sparsity. A potential explanation of the above discussed results might be that many latent features capture a very small number of instances (e.g., 1 or 2) which would lead to a large number of features with low label entropy. Such features would largely be useless as they make generalization very difficult. To verify that this is not the case, Figure 8 plots the label entropy versus the number of groundings of a predicate. If latent features of

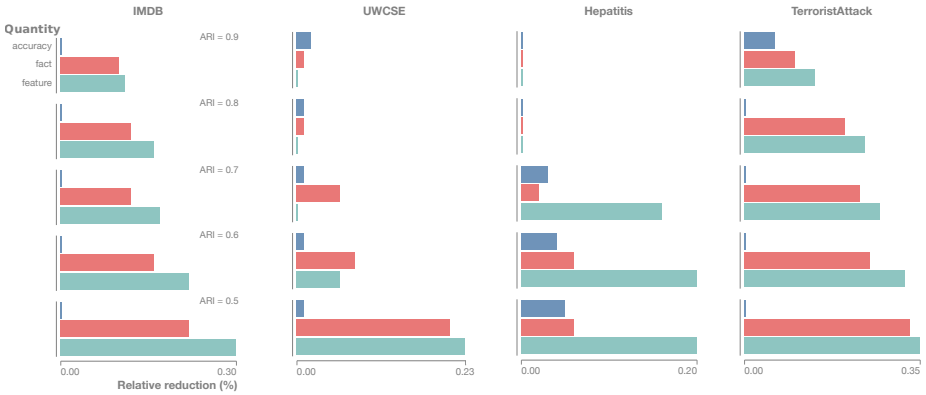


Fig. 9. The performance in terms of the accuracy is barely effected by removing overlapping clusterings, while the number of predicates and facts can be reduced up to 30%. The only noticeable reduction in performance happen on the Hepatitis dataset, but only for approximately 5%.

low label entropy would indeed capture only a small number of instances, many points would be condensed in the bottom left corner of the plot. However, that is not the case – many latent predicates with low label entropy actually have a number of groundings comparable to the predicates in the original representation. The exception to this is again the TerroristAttacks dataset.

These results jointly point to the following conclusion: *latent features successfully identify local regions in the instance space that match well with the provided labels*. As a consequence, these local regions are easier to capture and represent.

Redundancy. Figure 9 summarizes the influence of α on the accuracy and the number of latent features. The figure shows relative reduction in the number of features (equal to the number of predicates), the number of facts and the accuracy with respect to the latent representation obtained without rejecting the overlapping clusterings. These results show that the performance of the classifier is not affected by removing features based on the overlap of clusterings they define. The performance of TILDE remains approximately the same, whereas the number of latent features is reduced by 20 to 30 %. As the number of features is directly related to the size of the search space of relational model (and thus the complexity of learning), this is an encouraging result indicating that the size of the search space can be naively reduced without sacrificing the performance.

4.4 Looking forward

The proposed experimental framework is only the first step towards understanding how latent representations can benefit relational learning methods. The interaction between label entropy and sparsity seems to play an important role, indicative of the benefit of a latent representation. On the other hand, the method

for extracting the meaning of the latent features and analysis of their redundancy are developed especially for CUR^{2}LED and might have a limited benefit for future approaches.

Understanding when learning latent representation is (not) beneficial is an important question for further research. Majority of tasks benefits from learning latent representations, but some, like the TerroristAttack dataset, do not. Though we cannot definitely explain why that is the case, we suspect that the reason might be that the features of instances contain the most relevant information while the structure is uninformative. In contrast, CUR^{2}LED is developed to exploit the rich structure in relational dataset and is thus not suited for the scenario where only the features are relevant.

Another important question is how this kind of insights connects to the embeddings to vector spaces. The analysis done in this work focuses on contrasting the properties of predicates and associated data of original and latent representation obtained by CUR^{2}LED. The embeddings to vector spaces replace the logical representation of data with points in the Euclidean space and are thus not amenable to this kind of analysis. However, similar kind of analysis for embedding spaces is currently missing in the literature. Further research towards combining relational and deep learning methods might greatly benefit from understanding up- and downsides of both directions of research, and developing new ideas that combine advantages of both.

5 Conclusion

In this work we closely inspect the properties of latent representations for relational data. We focus on relational latent representations created by clustering both instances and relations among them, introduced by CUR^{2}LED [10]. The first property we analyse is the interpretability of latent features. We introduce a simple method to explain the meaning of latent features, and show that they capture interesting and sensible properties. Second, we identify two properties of these latent representation that partially explain their usefulness – namely, the label entropy and sparsity. Using these two properties, we show that obtained latent features identify local regions in instance space that match well with the labels. Consequently, this explains why predictive model learned from latent features are less complex and often perform better than the model learned from the original features. Third, we show that that latent features tend to be redundant, and that 20 to 30 % of latent features can be discarded without sacrificing the performance of the classifier. This consequently reduces the search space for the relational models, and simplifies learning.

Acknowledgements

This research is supported by Research Fund KU Leuven (GOA/13/010) and FWO (G079416N).

References

1. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
2. Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.
3. Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.
4. Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *JOURNAL OF LOGIC PROGRAMMING*, 19(20):629–679, 1994.
5. Stephen Muggleton. Predicate invention and utilization. *J. Exp. Theor. Artif. Intell.*, 6:121–130, 1994.
6. Stephen H. Muggleton, Dianhuan Lin, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Machine Learning*, 100(1):49–73, Jul 2015.
7. Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In Wolfram Burgard and Dan Roth, editors, *AAAI*. AAAI Press, 2011.
8. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 809–816, New York, NY, USA, 2011. ACM.
9. Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.
10. Sebastijan Dumančić and Hendrik Blockeel. Clustering-based relational unsupervised representation learning with an explicit distributed representation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1631–1637, 2017.
11. Sebastijan Dumančić and Hendrik Blockeel. An expressive dissimilarity measure for relational clustering using neighbourhood trees. *Machine Learning*, 2017.
12. Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, August 2013.
13. Leslie C. Morey and Alan Agresti. The measurement of classification agreement: An adjustment to the rand statistic for chance agreement. *Educational and Psychological Measurement*, 44(1):33–37, 1984.
14. Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1–2):285 – 297, 1998.

Noname manuscript No.
(will be inserted by the editor)

An expressive dissimilarity measure for relational clustering using neighbourhood trees

Sebastijan Dumančić · Hendrik Blockeel

Received: date / Accepted: date

Abstract Clustering is an underspecified task: there are no universal criteria for what makes a good clustering. This is especially true for relational data, where similarity can be based on the features of individuals, the relationships between them, or a mix of both. Existing methods for relational clustering have strong and often implicit biases in this respect. In this paper, we introduce a novel dissimilarity measure for relational data. It is the first approach to incorporate a wide variety of types of similarity, including similarity of attributes, similarity of relational context, and proximity in a hypergraph. We experimentally evaluate the proposed dissimilarity measure on both clustering and classification tasks using data set of very different types. Considering the quality of the obtained clustering, the experiments demonstrate that (a) using this dissimilarity in standard clustering methods consistently gives good results, whereas other measures work well only on data sets that match their bias; and (b) on most data sets, the novel dissimilarity outperforms even the best among the existing ones. On the classification tasks, the proposed method outperforms the competitors on the majority of data sets, often by a large margin. Moreover, we show that learning the appropriate bias in an unsupervised way is a very challenging task, and that the existing methods offer a marginal gain compared to the proposed similarity method, and can even hurt performance. Finally, we show that the asymptotic complexity of the proposed dissimilarity measure is similar to the existing state-of-the-art approaches. The results confirm that the proposed dissimilarity measure is indeed versatile enough to capture relevant information, regardless of whether that comes from the attributes of vertices, their proximity, or connectedness of vertices, even without parameter tuning.

Keywords Relational learning · Clustering · Similarity of structured objects

1 Introduction

In relational learning, the data set contains instances with relationships between them. Standard learning methods typically assume data are i.i.d. (drawn independently from the same population) and ignore the information in these relationships. Relational learning methods

Department of Computer Science, KU Leuven
Celestijnenlaan 200A, Heverlee, Belgium
E-mail: {sebastijan.dumancic,hendrik.blockeel}@cs.kuleuven.be

do exploit that information, and this often results in better performance. Complex data, such as relational data, is ubiquitous to the modern world. Among the most notable examples are social networks, which typically consist of a network of people interacting with each other. Another example includes rich biological and chemical data that often contains many interaction between atoms, molecules or proteins. Finally, any data stored in the form of relational databases is essentially relational data. Much research in relational learning focuses on supervised learning (De Raedt, 2008) or probabilistic graphical models (Getoor and Taskar, 2007). Clustering, however, has received less attention in the relational context.

Clustering is an underspecified learning task: there is no universal criterion for what makes a good clustering, it is inherently subjective. This is known for i.i.d. data (Estivill-Castro, 2002), and even more true for relational data. Different methods for relational clustering have very different biases, which are often left implicit; for instance, some methods represent the relational information as a graph (which means they assume a single binary relation) and assume that similarity refers to proximity in the graph, whereas other methods that take the relational database stance assume the similarity comes from relationships objects participate in. Such strong implicit biases make use of a clustering algorithm difficult for a problem at hand, without a deep understanding of both the clustering method and the problem at hand.

In this paper, we propose a very versatile framework for clustering relational data that makes the underlying biases transparent to a user. It views a relational data set as a graph with typed vertices, typed edges, and attributes associated to the vertices. This view is very similar to the viewpoint of relational databases or predicate logic. The task we consider is clustering the vertices of one particular type. What distinguishes our approach from other approaches is that the concept of (dis)similarity used here is very broad. It can take into account attribute dissimilarity, dissimilarity of the relations an object participates in (including roles and multiplicity), dissimilarity of the neighbourhoods (in terms of attributes, relationships, or vertex identity), and interconnectivity or graph proximity of the objects being compared.

Consider for example Figure 1. This relational dataset describes people and organizations, and relationships between them (friendship, a persons' role in the organization, ...). Persons and organizations are vertices in the graph shown there (shown as white/gray ellipses), the relationships between them are shown as edges, and their attributes are shown in dashed boxes. Now, vertices can be clustered in very different ways:

1. `Google` and `Microsoft` are similar because of their attributes, and could be clustered together for that reason
2. `John`, `Rose` and `Han` form a densely interconnected cluster
3. `Bob`, `Joe` and `Rose` share the property that they fulfill the role of supervisor

Non-relational clustering systems will yield clusters such as the first one; they only look at the attributes of individuals. Graph partitioning systems yield clusters of the second type. Some relational clustering systems yield clusters of the third type, which are defined by local structural properties. Most existing clustering systems have a very strong bias towards "their" type of clusters; a graph partitioning system, for instance, cannot possibly come up with the {Google, Microsoft} cluster, since this is not a connected component in the graph. The new clustering approach we propose is able to find all types of clusters, and even clusters that can only be found by mixing the biases.

The clustering approach and the corresponding dissimilarity measure that we propose are introduced in Section 2. Section 3 compares our approach to related work. Section 4

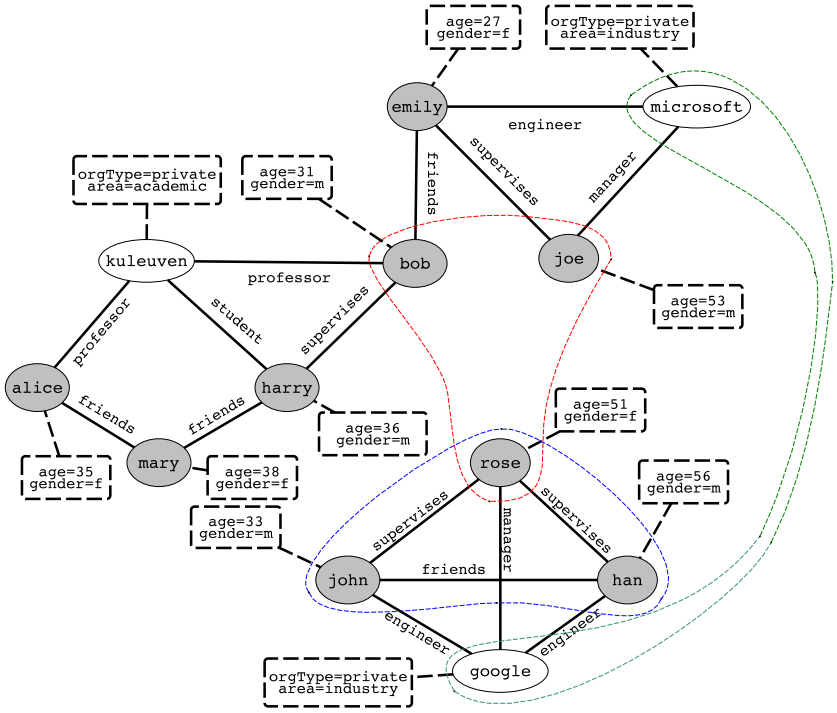


Fig. 1: An illustration of a relational data set containing people and organizations, and different clusters one might find in it. Instances - people and organizations, are represented by vertices, while relationships among them are represented with edges. The rectangles list an associated set of attributes for the corresponding vertex.

evaluates the approach, both from the point of view of clustering (the main goal of this work) as from the point of view of the dissimilarity measure introduced here (which can be useful also for, e.g., nearest neighbor classification). Section 5 presents conclusions.

2 Relational clustering over neighbourhood trees

2.1 Hypergraph Representation

Within relational learning, at least three different paradigms exist: inductive logic programming (Muggleton and De Raedt, 1994), which uses first-order logic representations; relational data mining (Dzeroski and Blockeel, 2004), which is set in the context of relational databases; and graph mining (Cook and Holder, 2006), where relational data are represented as graphs. We illustrate the different types of representation in Figure 2. This example represents a set of people and organizations, and relationships between them. The relational database format *(b)* is perhaps the most familiar to most people. It has a table for each entity type (`Person, Organization`) and for each relationship type between entities (`Works_for, Friends`). Each table contains multiple attributes, each of which can be an identifier for a particular entity (a *key attribute*, e.g., `PName`), or a property of that en-

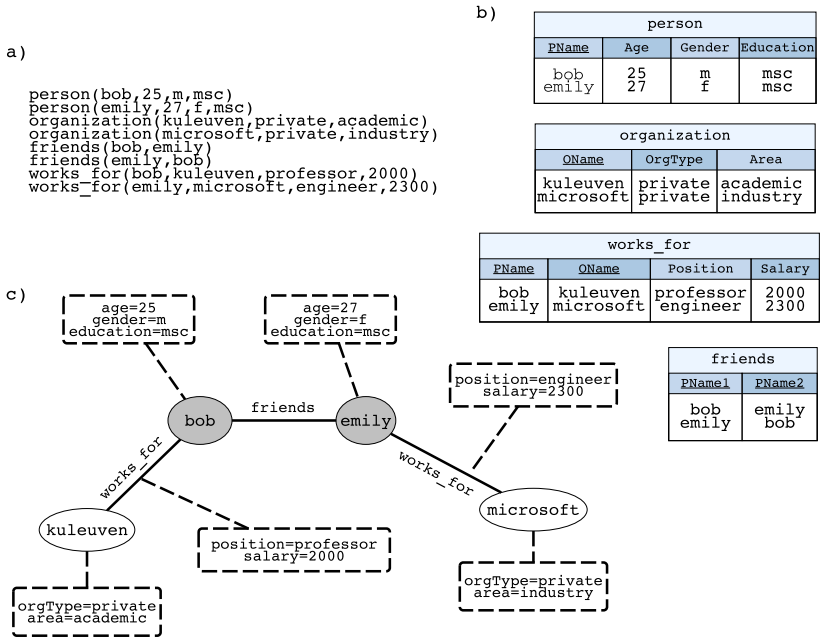


Fig. 2: Representation paradigms of relational data. Section *a)* represents the relational data set as a set of logical facts; the upper part represents the definition of each predicate, while the bottom part lists all facts. Section *b)* illustrates a *database view* of the relational data set, where each logical predicate is associated with a single database table. Section *c)* illustrates a *graph view* of the relational data set. Each circle represents an instance, each rectangle represents attributes associated with the corresponding instance, while relations are represented by the edges.

tity (`Age,Gender,...`). The logic-based format *(a)* is very similar; it consists of logical facts, where the predicate name corresponds to the tables name and the arguments to the attribute values. There is a one-to-one mapping between rows in a table and logical facts. The logic based view allows for easy integration of background knowledge (in the form of first-order logic rules) with the data. Finally, there is the attributed graph representation *(c)*, where entities are nodes in the graph, binary relationships between them are edges, and nods and edges can have attributes. This representation has the advantage that it makes the entities and their connectivity more explicit, and it naturally separates identifiers from real attributes (e.g., the `PName` attribute from the `Person` table is not listed as an attribute of `Person` nodes, because it only serves to uniquely identify a person, and in the graph representation the node itself performs that function). A disadvantage is that edges in a graph can represent only binary relationships.

Though the different representations are largely equivalent, they provide different views on the data, which affects the clustering methods used. For instance, a notion such as shortest path distance is much more natural in the graph view than in the logic based view, while the fact that there are different types of entities is more explicit in the database view (one table per type). The distinction between entities and attribute values is explicit in the graph, but more implicit in the database view (key vs. non-key attributes) and absent in the logic view.

In this paper, we will use a hypergraph view that combines elements of all the above. An oriented hypergraph is a structure $H = (V,E)$ where V is a set of vertices and E a set of hyperedges; a hyperedge is an ordered multiset whose elements are in V. Directed graphs are a special case of oriented hypergraphs where all hyperedges have cardinality two.

A set of relational data is represented by a typed, labeled, oriented hypergraph (V,E,τ,λ) with V a set of vertices, E a set of hyperedges, and $\tau : (V \cup E) \to T_V \cup T_E$ a type function that assigns a type to each vertex and hyperedge (T_V is the set of vertex types, T_E the set of hyperedge types). With each type $t \in T_V$ a set of attributes $A(t)$ is associated, and λ maps each vertex v to a vector of values, one value for each attribute in $A(\tau(v))$. If $a \in A(\tau(v))$, we write $a(v)$ for the value of a in v.

A relational database can be converted into the hypergraph representation as follows.[1] For each table with only one key attribute (describing the entities identified by that key), a vertex type is introduced, whose attributes are the non-key attributes of the table. Each row becomes one vertex, whose identifier is the key value and whose attribute values are the non-key attribute values in the row. For each table with more than one key attribute (describing non-unary relationships among entities), a hyperedge is introduced that contains the vertices corresponding to these entities in the order they occur in the table. Our hypergraph representation does not associate attributes with hyperedges, only with vertices; hence, for non-unary relationships contain non-key attributes, a new vertex type corresponding to that hyperedge type is introduced.

The clustering task we consider is the following: given a vertex type $t \in T_V$, partition the vertices of this type into clusters such that vertices in the same cluster tend to be similar, and vertices in different clusters dissimilar, for some subjective notion of similarity. In practice, it is of course not possible to use a subjective notion; one uses a well-defined similarity function, which hopefully on average approximates well the subjective notion that the user has in mind. The following section introduces *neighbourhood trees*, a structure we use to compactly represent and describe a neighbourhood of a vertex.

2.2 Neighbourhood tree

A neighbourhood tree is a directed graph rooted in a vertex of interest, i.e. the vertex whose neighbourhood one wants to describe. It is constructed simply by following the hyperedges from the root vertex, as outlined in Algorithm 1. The construction of the neighbourhood tree is parametrized with the pre-specified depth, a vertex of interest and the original hypergraph. Consider a vertex v. For every hyperedge E in which v participates (lines 7-13), add a directed edge from v to each vertex $v' \in E$ (line 9). Label each vertex with its type, and attach to it the corresponding attribute vector (line 10). Label the edge with the hyperedge type and the position of v in the hyperedge (recall that hyperedges are ordered sets; line 11). The vertices thus added are said to be at depth 1. If there are multiple hyperedges connecting vertices v and v', v' is added each time it is encountered. Repeat this procedure for each v' on depth 1 (stored in variable `toVisit`). The vertices thus added are at depth 2. Continue this procedure up to some predefined depth d. The root element is never added to the subsequent levels. An example of a neighbourhood tree is given in Figure 3.

The following section introduces a dissimilarity measure for vertices of the hypergraph.

[1] For the logic-based representation, the conversion is analogous.

Algorithm 1: Neighbourhood tree construction

```
   Data: a hypergraph H = (V, E, τ, λ)
         a vertex of interest v
         a depth d
   Result: a neighbourhood tree NT
   /* initialize neighbourhood tree                                  */
 1 NT = new neighbourhood tree;
 2 NT.addRoot(v);
 3 NT.labelVertex(v) ;                        /* add type and attributes */
 4 toVisit = {v} ;                              /* vertices to process */
 5 d' = 1 ;                                        /* depth indicator */
   /* repeat until the pre-specified depth                           */
 6 while d' ≤ d do
 7     foreach v' ∈ toVisit do
 8         foreach outgoing edge e of vertex v' do
 9             foreach vertex v'' in hyperedge e do
10                 NT.addVertex(v'');
11                 NT.addEdge(v',v'');
12                 NT.labelVertex(v'') ;          /* add type and attributes */
13                 NT.labelEdge(v', v'') ;      /* add edge type and position */
14                 toVisit = toVisit ∪ {v''};
15             end
16         end
17         toVisit = toVisit \ {v',v}
18     end
19     d' += 1;
20 end
```

2.3 Dissimilarity measure

The main idea behind the proposed dissimilarity measure is to express a wide range of similarity biases that can emerge in relational data, as discussed and exemplified in Section 1. The proposed dissimilarity measure compares two vertices by comparing their neighbourhood trees. It does this by comparing, for each level of the tree, the distribution of vertices, attribute values, and outgoing edge labels observed on that level. Earlier work in relational learning has shown that distributions are a good way of summarizing neighbourhoods (Perlich and Provost, 2006).

The method for comparing distributions distinguishes between discrete and continuous domains. For discrete domains (vertices, edge types, and discrete attributes), the distribution simply maps each value to its relative frequency in the observed multiset of values, and the χ^2-measure for comparing distributions (Zhao et al, 2011) is used. That is, given two multisets A and B, their dissimilarity is defined as

$$d(A,B) = \sum_{x \in A \cup B} \frac{(f_A(x) - f_B(x))^2}{f_A(x) + f_B(x)} \tag{1}$$

where $f_S(x)$ is the relative frequency of element x in multiset S (e.g., for $A = \{a,b,b,c\}$, $f_A(a) = 0.25$ and $f_A(b) = 0.5$).

In the continuous case, we compare distributions by applying aggregate functions to the multiset of values, and comparing these aggregates. Given a set $\mathscr{A}$ of aggregate functions,

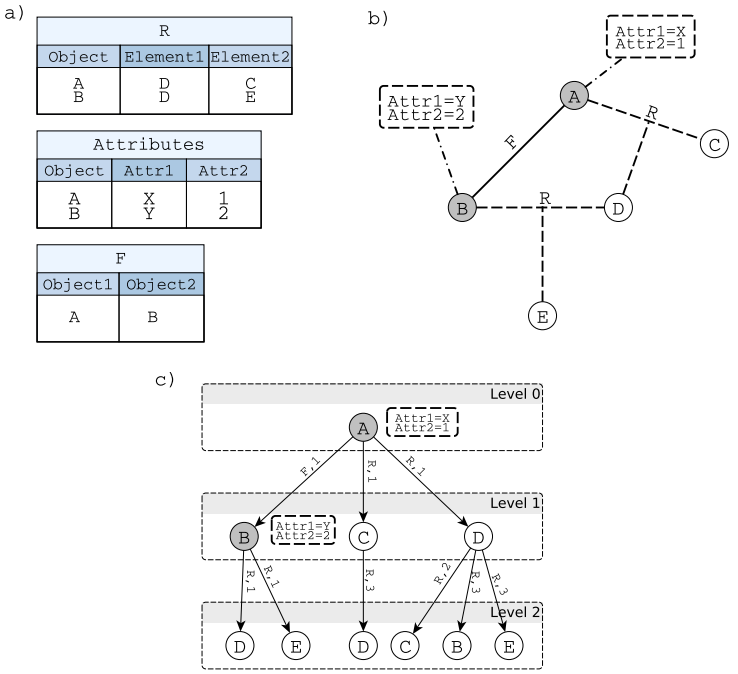


Fig. 3: An illustration of the neighbourhood tree. The domain contains two types of vertices - *objects* (A and B) and *elements* (C, D and E), and two fictitious relations: R and F. The vertices of type *object* have an associated set of attributes. Section *a)* contains the database view of the domain. Section *b)* contains the corresponding hypergraph view. Here, edges are represented with full lines, while hyperegdges are represented with dashed lines. Finally, section *c)* contains the corresponding *neighbourhood tree* for the vertex A.

the dissimilarity is defined as

$$d(A,B) = \sum_{f \in \mathscr{A}} \frac{f(A) - f(B)}{r} \tag{2}$$

with r a normalization constant ($r = \max_M f(M) - \min_M f(M)$, with M ranging over all multisets for this attribute observed in the entire set of neighbourhood trees). In our implementation, we use the mean and standard deviation as aggregate functions.

The above methods for comparing distributions have been chosen for their simplicity and ease of implementation. More sophisticated methods could be used. The main point of this section, however, is *which* distributions are compared, not *how* they are compared.

We use the following notation. For any neighbourhood tree g,

- $V^l(g)$ is the multiset of vertices at depth l (the root having depth 0)
- $V^l_t(g)$ is the multiset of vertices of type t at depth l
- $B^l_{t,a}(g)$ is the multiset of values of attribute a observed among the nodes of type t at depth l
- $E^l(g)$ is the multiset of edge types between depth l and $l+1$

E.g., for the neighbourhood tree in Figure 3, we have

- $V^1(g) = \{\texttt{B, C, D}\}$
- $V^1_{object}(g) = \{\texttt{B}\}$
- $E^1(g) = \{(\texttt{F, 1}), (\texttt{R, 1}), (\texttt{R, 1})\}$
- $B^1_{object,Attr1}(g) = \{\texttt{Y}\}$

Let $\mathscr{N}$ be the set of all neighbourhood trees corresponding to the vertices of interest in a hypergraph. Let $norm(\cdot)$ be a *normalization operator*, defined as

$$norm(f(g_1,g_2)) = \frac{f(g_1,g_2)}{\max\limits_{g,g' \in \mathscr{N}} f(g,g')},$$

i.e., the normalization operator divides the value of the function $f(g_1,g_2)$ of two neighbourhood trees g_1 and g_2 by the highest value of the function f obtained amongst all pairs of neighbourhood trees.

Intuitively, the proposed method starts by comparing two vertices according to their attributes. It then proceeds by comparing the properties of their neighbourhoods: which vertices are in there, which attributes they have and how are they interacting. Finally, it looks at the proximity of vertices in a given hypergraph. Formally, the dissimilarity of two vertices v and v' is defined as the dissimilarity of their neighbourhood trees g and g', which is:

$$\begin{aligned} s(g,g) = & w_1 \cdot \mathsf{ad}(g,g) + w_2 \cdot \mathsf{nad}(g,g) + w_3 \cdot \mathsf{cd}(g,g) \\ & + w_4 \cdot \mathsf{nd}(g,g) + w_5 \cdot \mathsf{ed}(g,g) \end{aligned} \tag{3}$$

where $\sum_i w_i = 1$ and

– *attribute-wise dissimilarity*

$$\mathsf{ad}(g,g') = norm\left(\sum_{a \in A(\tau(v))} d(B^0_{t,a}(g), B^0_{t,a}(g'))\right) \tag{4}$$

measures the dissimilarity of the root elements v and v' according to their attribute-value pairs.

– *neighbourhood attribute dissimilarity*

$$\mathsf{nad}(g,g') = norm\left(\sum_{l=1}^{d} \sum_{t \in T_V} \sum_{a \in A(t)} d(B^l_{t,a}(g), B^l_{t,a}(g'))\right) \tag{5}$$

measures the dissimilarity of attribute-value pairs associated with the neighbouring vertices of the root elements, per level and vertex type.

– *connection dissimilarity*

$$\mathsf{cd}(g,g') = 1 - norm\left(|\{v \in V^0(g) | v \in V^1(g')\}|\right) \tag{6}$$

reflects the number of edges of different type that exist between the two root elements.

– *neighbourhood dissimilarity*

$$\mathsf{nd}(g,g') = norm\left(\sum_{l=1}^{\#levels}\sum_{t\in T_v} d(V_t^l(g),V_t^l(g'))\right) \tag{7}$$

measures the dissimilarity of two root elements according to the vertex identities in their neighbourhoods, per level and vertex type.

– *edge distribution dissimilarity*:

$$\mathsf{ed}(g,g) = norm\left(\sum_{l=1}^{\#levels} d(E^l(g),E^l(g))\right) \tag{8}$$

measures the dissimilarity over edge types present in the neighbourhood trees, per level.

Each component is normalized to the scale of 0-1 by the highest value obtained amongst all pair of vertices, ensuring that the influence of each factor is proportional to its weight. The weights w_{1-5} in Equation 3 allow one to formulate a bias through the similarity measure. For the remainder of the text, we will term our approach as ReCeNT (for Relational Clustering using Neighbourhood Trees). The benefits and downsides of this formulation are discussed and contrasted to the existing approaches in Sections 3.3 and 4.3.

This formulation is somewhat similar to the *multi-view clustering* (Bickel and Scheffer, 2004), with each of the components forming a different view on data. However, there is one important fundamental difference: multi-view clustering methods want to find clusters that are good in each view separately, whereas our components do not represent different views on the data, but different potential biases, which jointly contribute to the similarity measure.

3 Related work

3.1 Hypergraph representation

Two interpretations of the hypergraph view of relational data exist in literature. The one we incorporate here, where domain objects form vertices in a hypergraph with associated attributes, and their relationships form hyperedges, was first introduced by Richards and Mooney (1992). An alternative view, where logical facts form vertices, is presented by Ong et al (2005). Such representations were later used to learn the formulas of relational models by *relational path-finding* (Kok and Domingos, 2010; Richards and Mooney, 1992; Ong et al, 2005; Lovász, 1996).

The neighbourhood tree introduced in Section 2.2 can be seen as summary of all paths in a hypergraph originating at a certain vertex. Though neighbourhood trees and relational path-finding rely on a hypergraph view, the tasks they solve are conceptually different. Whereas the goal of the neighbourhood tree is to compactly represent a neighbourhood of a vertex by summarizing all the paths originating at the vertex, the goal of relational path-finding is to identify a small set of important paths that appear often in a hypergraph. Additionally, a practical difference is the distinction between hyperedges and attributes - a neighbourhood tree is constructed by following only the hyperedges, while the mentioned work either treats attributes as unary hyperedges or requires a declarative bias from the user.

3.2 Related tasks

Two problems related to the one we consider here are graph and tree partitioning (Bader et al, 2013). Graph partitioning focuses on partitioning the original graph into a set of smaller graphs such that certain properties are satisfied. Though such partitions can be seen as clusters of vertices, the clusters are limited to vertices that are connected to each other. Thus, the problem we consider here is strictly more general, and does not put any restriction of that kind on the cluster memberships; the (dis)similarity of vertices can originate in any of the (dis)similarity sources we consider, most of which cannot be expressed within a graph partitioning problem.

A number of tree comparison techniques (Bille,2005) exists in the literature. These approaches consider only the identity of vertices as a source of similarity, while ignoring the attributes and types of both vertices and hyperedges. Thus, they are not well suited for the comparison of neighbourhood trees.

3.3 Relational clustering

The relational learning community, as well as the graph kernel community have previously shown interest in clustering relational (or structured) data. Existing similarity measures within the relational learning community can be coarsely divided into two groups.

The first group consists of similarity measures defined over an attributed graph model (Pfeiffer et al, 2014), with examples in *Hybrid similarity (HS)* (Neville et al, 2003) and *Hybrid similarity on Annotated Graphs (HSAG)* (Witsenburg and Blockeel, 2011). Both approaches focus on attribute-based similarity of vertices where HS compares the attributes of all connected vertices, and HSAG's similariy measure compares attributes of the vertices themselves and attributes of their neighbouring vertices. The main limitations of these approaches are that they ignore the existence of vertex and edge types, and impose a very strict bias towards attributes of vertices. In comparison to the presented approach, HS defines dissimilarity as the ad component if there is an edge between two vertices, and ∞ otherwise. HSAG defines the dissimilarity as a linear combination of the ad and nad components for each pair of vertices.

In contrast to the first group which employs a graph view, the second group of methods employs a predicate logic view, The two most prominent approaches are *Conceptual clustering of Multi-relational Data (CC)* (Fonseca et al, 2012) and *Relational instance-based learning* (RIBL) (Emde and Wettschereck, 1996; Kirsten and Wrobel, 1998). CC firstly describes each example (corresponding to a vertex in our problem) with a set of logical clauses that can be generated by a *bottom clause saturation* (Camacho et al, 2007). The obtained clauses are considered as features, and their similarity is measured by the *Tanimoto similarity* - a measure of overlap between sets. In that sense, it is similar to using the ad and ed components for generating clauses. Note that this approach does not differentiate between relations (or interactions) and attributes, does not consider distributions of any kind, and does not have a sense of depth of a neighbourhood. Finally, RIBL follows an intuition that the similarity of two objects depends on the similarity of their attributes' values and the similarity of the objects related to them. To that extent, it first constructs a *context descriptor* - a set of objects related to the object of interest, similarly to the neighbourhood trees. Comparing two object now involves comparing their features and computing the similarity of the set of objects they are linked to. That requires matching each object of one set to the

Table 1: Aspects of similarity considered by different approaches. ✓ denotes full consideration, ≃ partial and × no consideration at all.

Similarity	Attributes	Neighbourhood attributes	Neighbourhood identities	Proximity	Structural properties
ReCeNT	✓	✓	✓	✓	✓
HS	✓	×	×	×	×
HSAG	✓	✓	×	×	×
RIBL	✓	✓	✓	×	×
CC	≃	≃	×	×	≃
RKOH	×	≃	×	×	✓
WLST	×	≃	×	×	✓

most similar object in the other set, which is an expensive operation (proportional to the product of the set sizes). In contrast, the χ^2 distance is linear in the size of the multiset. Further, the χ^2 distance takes the multiplicity of elements into account (it essentially compares distributions), which the RIBL approach does not.

Within the graph kernel community, two prominent groups exist: *Weisfeiler-Lehman graph kernels (WL)* (Shervashidze et al, 2011; Shervashidze and Borgwardt, 2009; Frasconi et al, 2014; Haussler, 1999; Bai et al, 2014) and *random walk based kernels* (Wachman and Khardon, 2007; Lovász, 1996). A common feature of these approaches is that they measure a similarity of graph by comparing their structural properties. The Weisfeiler-Lehman Graph Kernels is a family of graph kernels developed upon the *Weisfeiler-Lehman isomorphism test*. The key idea of the WL isomorphism test is to extend the set of vertex attributes by the attributes of the set of neighbouring vertices, and compress the augmented attribute set into new set of attributes. There each new attribute of a vertex corresponds to a subtree rooted from the vertex, similarly to the neighbourhood trees. Shervashidze and Borgwardt have introduced a fast WL subtree kernel (WLST) (Shervashidze and Borgwardt, 2009) for undirected graphs by performing the WL isomorphism test to update the vertex labels, followed by counting the number of matched vertex labels. The difference between our approach and WL kernel family is subtle but important: WL graph kernels extend the set of attributes by identifying isomorphic subtrees present in (sub)graphs. This is reflected in the bias they impose, that is, the similarity comes from the structure of a graph (in our case, a neighbourhood tree).

A Rooted Kernel for Ordered Hypergraph (RKOH) (Wachman and Khardon, 2007]) is an instance of random walk kernels successfully applied in relational learning tasks. These approaches estimate the similarity of two (hyper)graphs by comparing the walks one can obtain by traversing the hypergraph. RKOH defines a similarity measure that compares two hypergraphs by comparing the paths originating at every edge of both hypergraphs, instead of the paths originating at the root of the hypergraph. RKOH does not differentiate between attributes and hyperedges, but treats everything as an hyperedge instead (an attribute can be seen as an unary edge).

Table 1 summarizes different aspects of similarity considered by the above mentioned approaches. The interpretations of similarity are divided into five sources of similarity. The first two categories concern attributes: attributes of the vertices themselves and their neighbouring vertices. The following two categories concern identities of vertices in the neighbourhood of a vertex of interest. They concern subgraphs (identity of vertices in the neighbourhood) centered at a vertex, and proximity of two vertices. The final category concerns

Table 2: Complexities of different approaches

Approach	Complexity
HS	$O(LA)$
HSAG	$O\left(N^2EA\right)$
ReCeNT	$O\left(N^2E^d\right)$
WLST	$O\left(N^2E^d\right)$
CC	$O\left(N^2\binom{E+A}{l}\right)$
RIBL	$O\left(N^2\prod_{k=1}^{d}(E+A)^{2k}\right)$
RKOH	$O\left(N^2(E+A)^{2d+2l}\right)$

the structural properties of subgraphs in the neighbourhood of a vertex defined by the neighbourhood tree.

3.3.1 Complexity analysis

Though scalability is not the focus of this work, here we show that the proposed approach is as scalable as the state–of–the–art kernel approaches, and substantially less complex than the majority of the above-mentioned approaches that use both attribute and link structure. For the sake of clarity of comparison, assume a homogeneous graph with only one vertex type and one edge type. Let N be the number of vertices in a hyper-graph, L be the total number of hyperedges, and d be the depth of a neighbourhood representation structure, where applicable. Let, as well, A be the number of attributes in a data set. Additionally, assume that all vertices participate in the same number of hyperedges, which we will refer to as E. We will refer to the length of clause in CC and path in RKOH as l.

To compare any two vertices, ReCeNT requires one to compute the dissimilarity of the multisets representing the vertices, proportional to $O(d \times A + \sum_{k=1}^{d} E^k) = O\left(N^2E^d\right)$. Table 2 summarizes the complexities of the discussed approaches. In summary, the approaches can be grouped into three categories. The first category contains HS and HSAG; these are substantially less complex than the rest, but focus only on the attribute similarities. The second category contains RIBL and RKOH, which are substantially more complex than the rest. Both of these approaches use both attribute and edge information, but in a computationally very expensive way. The last category contains ReCeNT, WLST and CC; these lie in between. They use both attribute and edge information, but in a way that is much more efficient than RIBL and RKOH.

The complexity of ReCeNT benefits mostly from two design choices: *differentiation of attributes and hyperedges*, and *decomposition of neighbourhood elements into multisets*. By distinguishing hyperedges from attributes, ReCeNT focuses on identifying sparse neighbourhoods. Decomposition of neighbourhoods into multisets allows ReCeNT to compute the similarity linearly in the size of a multiset. The parameter that ReCeNT is the most sensitive to is the depth of the neighbourhood tree, which is the case with the state–of–the–art kernel approaches as well. However, the main underlying assumption behind ReCeNT is that important information is contained in small local neighbourhoods, and ReCeNT is designed to use such information.

Table 3: Characteristics of the data sets used in experiments. The characteristics include the total number of vertices, the number of vertices of interest, the total number of attributes, the number of attributes associated with vertices of interest, the number of hyperedges as well as the number of different hyperedge types.

data set	IMDB	UW–CSE	Muta	WebKB	Terror
#vertices	298	734	6124	3880	1293
#target vertices	268	272	230	920	1293
#vertex types	3	4	2	2	1
#attributes	3	7	7	1207	106
#target attributes	3	3	4	763	106
#hyperedges	715	1834	30804	5779	3743
#hyperedge types	3	6	7	4	2

4 Evaluation

4.1 Data sets

We evaluate our approach on five data sets for relational clustering with different characteristics and domains. The chosen data sets are commonly used within the (statistical) relational learning community, and they expose different biases. The characterization of data sets, summarized in Table 3, include the total number of vertices in a hypergraph, the number of vertices of interest, the total number of attributes, the number of attributes associated with vertices of interest, the number of hyperedges as well as the number of different hyperedge types. The data sets range from having a small number of vertices, attributes and hyperedges (UW–CSE, IMDB), to a considerably large number of vertices, attributes or hyperedges (Mutagenesis, WebKB, TerroristAttack). All the chosen data sets are originally classification data sets, which allows us to evaluate our approach with respect to how well it extracts the classes present in the data set.

The IMDB[2] data set is a small snapshot of the Internet Movie Database. It describes a set of movies with people acting in or directing them. The goal is to differentiate people into two groups: *actors* and *directors*. The UW–CSE[3] data set describes the interactions of employees at the University of Washington and their roles, publications and the courses they teach. The task is to identify two clusters of people: *students* and *professors*. The Mutagenesis[4] data set describes chemical compounds and atoms they consist of. Both compounds and atoms are described with a set of attributes describing their chemical properties. The task is to identify two clusters of compounds: *mutagenic* and *not mutagenic*. The WebKB[5] data set consists of pages and links collected from the Cornell University's webpage. Both pages and links are associated with a set of words appearing on a page or in the anchor text of a link. The pages are classified into seven groups according to their role, such as *personal*, *departmental* or *project* page. The final data set, termed Terrorists[6] [Sen et al,2008], describes terrorist attacks each assigned one of 6 labels indicating the type of the attack. Each attack is described by a total of 106 distinct features, and two relations indicating whether two attacks were performed by the same organization or at the same location.

[2] Available at http://alchemy.cs.washington.edu/data/imdb

[3] Available at http://alchemy.cs.washington.edu/data/uw-cse/

[4] Available at http://www.cs.ox.ac.uk/activities/machlearn/mutagenesis.html

[5] Available at http://alchemy.cs.washington.edu/data/webkb/

[6] Available at http://linqs.umiacs.umd.edu/projects//projects/lbc/

4.2 Experiment setup

In the remainder of this section, we evaluate our approach. We focus on answering the following questions:

(Q1) *How well does ReCeNT perform on the relational clustering tasks compared to existing similarity measures?*
(Q2) *How relevant is each of the components?* We perform clustering using our similarity measure and setting the parameters as $w_i = 1, w_{j,j\neq i} = 0$.
(Q3) *To which extent can the parameters of the proposed similarity measure be learnt from data in an unsupervised manner?*
(Q4) *How well does ReCeNT perform compared to existing similarity measures in a supervised setting?*
(Q5) *How do the runtimes for ReCeNT compare to the competitors?*

In each experiment, we have used the aforementioned (dis)similarity measures in conjunction with spectral [Ng et al, 2001] and hierarchical [Ward, 1963] clustering algorithms, as implemented in `scikit-learn` [Pedregosa et al, 2011][7]. We have intentionally chosen two clustering approaches which assume different biases, to be able to see how each similarity measure is affected by assumptions clustering algorithms make. We have altered the depth on neighbourhood trees between 1 and 2 wherever it was possible, and report both results.

We evaluate each approach using the following validation method: we set the number of clusters to be equal to the true number of clusters in each data set, and evaluate the obtained clustering with regards to how well it matches the known clustering given by the labels. Each obtained clustering is then evaluated using the *adjusted Rand index* (ARI) [Rand, 1971; Morey and Agresti, 1984]. The ARI measures the similarity between two clusterings, in this case between the obtained clustering and the provided labels. The ARI score ranges between -1 and 1, where a score closer to 1 corresponds to higher similarity between two clusterings, and hence better performance, while 0 is the chance level. For each data set, and each similarity measure, we report the ARI score they achieve. Additionally, we have set a timeout to 24 hours and do not report results for an approach that takes more time to compute.

To achieve a fair time comparison, we implemented all similarity measures (HS, HSAG, RIBL, CC, as well as RKOH) in `Scala` and optimized them in the same way, by caching all the intermediate results that can be re-used. The hierarchy obtained by hierarchical clustering was cut when it has reached the pre-specified number of clusters. In the first experiment, the weights w_{1-5} were not tuned, and were set to 0.2. We have used mean and standard deviation as aggregates for continuous attributes.

4.3 Results

4.3.1 ***(Q1) Comparison to the existing methods***

Using exactly the same clustering algorithms, we compare ReCeNT to a variety of (dis)similarity measures: a baseline approach using the Euclidean distance between attributes of vertices and no relationships; HS [Neville et al, 2003], HSAG [Witsenburg and Blockeel, 2011],

[7] more precisely, sklearn.cluster.SpectralClustering and sklearn.cluster.AgglomerativeClustering

Table 4: Performance of all approaches on three data sets. For each similarity measure, the ARI achieved when the true number of clusters was used. The results are shown for both hierarchical and spectral clustering,while the depth of the approaches is indicated by the subscript. The last column counts the number of wins per algorithm, where "win" means achieving the highest ARI on a data set.

Similarity	**Muta**		**UWCSE**		**WebKB**		**Terror**		**IMDB**		**W**
	H	S	H	S	H	S	H	S	H	S	
Baseline	-0.02	-0.03	0.25	0.2	0.00	0.25	0.00	0.17	0.05	0.05	0
HS	N/A	N/A	0.01	0.06	0.0	0.10	0.01	-0.01	0.00	0.00	0
CC$_2$	0.00	0.01	0.1	0.82	0.00	0.04	0.01	0.01	0.1	0.1	0
CC$_4$	0.00	0.01	0.00	0.92	0.00	0.04	0.01	0.01	0.1	0.1	0
ReCeNT$_1$	**0.32**	**0.35**	**0.97**	**0.98**	**0.04**	**0.57**	0.00	**0.26**	0.62	**1.0**	**8**
RIBL$_1$	0.22	0.26	0.89	0.68	0.0	0.1	N/A	N/A	0.35	0.38	0
HSAG$_1$	-0.01	0.06	0.1	0.0	0.01	0.05	0.00	0.24	0.04	-0.05	0
WLST$_{1,5}$	0.00	0.02	-0.01	0.33	0.00	0.33	**0.27**	0.07	-0.01	0.66	1
WLST$_{1,10}$	0.00	0.02	-0.01	0.33	0.00	0.32	**0.27**	0.11	-0.01	0.31	1
V$_1$	0.00	0.03	-0.01	0.19	0.00	0.00	0.00	0.00	0.00	0.00	0
VE$_1$	0.00	0.03	0.01	0.36	0.00	0.00	0.00	0.00	**1.0**	**1.0**	2
RKOH$_{1,2}$	0.1	0.1	0.2	0.2	N/A	N/A	N/A	N/A	0.83	0.83	0
RKOH$_{1,4}$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0
ReCeNT$_2$	0.08	0.3	0.1	0.16	0.02	0.4	0.01	0.16	0.13	**1.0**	1
RIBL$_2$	N/A	N/A	0.0	0.68	N/A	N/A	N/A	N/A	0.63	0.78	0
HSAG$_2$	-0.01	0.06	0.1	0.0	0.0	0.04	0.00	0.23	0.04	0.09	0
WLST$_{2,5}$	0.00	0.01	0.02	0.02	0.00	0.52	**0.27**	0.11	-0.04	0.31	1
WLST$_{2,10}$	0.00	0.01	0.02	0.02	0.00	0.52	0.05	0.12	-0.04	0.36	0
V$_2$	0.00	0.07	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
VE$_2$	0.00	0.00	0.01	0.38	0.00	**0.56**	0.00	0.00	0.00	0.53	1
RKOH$_{2,2}$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0
RKOH$_{2,4}$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0

CC [Fonseca et al, 2012], RIBL [Emde and Wettschereck, 1996], as well as Weisfeiler-Lehman subtree kernel (WLST) [Shervashidze and Borgwardt, 2009], Linear kernel between vertex histograms (V), Linear kernel between vertex-edge histograms (VE) provided with [Sugiyama and Borgwardt,2015], and RKOH [Wachman and Khardon, 2007]. The subscript in ReCeNT, HSAG, RIBL and kernel approaches denotes the depth of the neighbourhood tree (or other supporting structure). The subscript in CC denotes the length of the clauses. The second subscript in WLST and RKOH indicates their parameters: with WLST it is the *h* parameter indicating the number of iterations, whereas with RKOH it indicates the length of the walk.

The results of the first experiment are summarized in Table 4. The table contains ARI values obtained by the similarity measures for each data set and clustering algorithm used. The last column of the table states the number of wins per approach. The number of wins is calculated by simply counting the number of cases where the approach obtained the highest ARI value, a "case" being a combination of a data set and a clustering algorithm. ReCeNT$_1$ wins 8 out of 10 times, and thus outperforms all other methods. The best results are achieved in combination with spectral clustering, with exception being the TerroristAttack data set where WLST$_{1,*}$ and WLST$_{2,5}$ combined with hierarchical clustering achieved the highest ARI of 0.27, in contrast to 0.26 obtained by ReCeNT$_1$. In all cases of the Mutagenesis and UWCSE data sets, ReCeNT$_1$ wins with a larger margin. However, it is important to note that

in the remaining cases, the closest competitor is not always the same. In the case of IMDB data set in combination with spectral clustering, the closest competitor is VE_1 (together with $RKOH_{1,2}$), as well as in the case of WebKB in combination with spectral clustering. In the cases of the TerroristAttack data set combined with the spectral clustering, the closest competitors are $HSAG_1$ and $HSAG_2$, while in the case with hierarchical clustering our approach is outperformed by $WLST_{1,*}$ and $WLST_{2,5}$. These results show that the proposed similarity measure performs better over wide range of different tasks and biases, compared to the remaining approaches. Moreover, when combined with the spectral clustering, $ReCeNT_1$ consistently performs well on all data sets, achieving the second-best result only on the TerroristAttack data set.

Each of the data sets exposes different bias, which influences the performance of the methods. In order to successfully identify mutagenic compounds, one has to consider both attribute and link information, including the attributes of the neighbours. Chemical compounds that have similar structure tend to have similar properties. This data set is more suitable for RIBL, ReCeNT and kernel approaches. $ReCeNT_1$ and $RIBL_1$ achieve the best results here[8], while kernels approaches surprisingly do not perform better than the chance level. The UW-CSE is a social-network-like data set where the task is to find two interacting communities with different attribute-values - students and professors. The distinction between two classes is made on a single attribute – professors have positions, while students do not, and the relation stating that professors advise students. This task is suitable for HS and HSAG. However, both approaches are substantially outperformed by $ReCeNT_1$ and CC_*. Similarly, the IMDB data set consists of a network of people and their roles in movies, which can be seen as a social network. Here, directors can be differentiated from actors by a single edge type – actors work under directors which is explicitly encoded in the data set. The type of interactions between entities matters the most, as it is not an attribute-rich data set, and is thus more suitable for methods that account for structural measures. Accordingly, ReCeNT, RIBL, $WLST_{1,*}$ and VE kernels achieve the best results.

The remaining data sets, WebKB and TerroristAttack, are entirely different in nature from the aforementioned ones. These data set have a substantially larger number of attributes, but those are not sufficient to identify relevant clusters supported by labels, that is, interactions contain important information. Such bias is implicitly present in HS, and partially assumed by kernel approaches. The results show that $ReCeNT_1$ and $WSLT_{2,*}$ and VE_2 kernels achieve almost identical performance on the WebKB data set, while the remaining approaches are outperformed even by the baseline approach. On the TerroristAttack data set, $WLST_{1,*}$ kernel achieves the best performance, outperforming $ReCeNT_1$ and $HSAG_1$. Similarly to WebKB, other approaches are outperformed by the baseline approach.

The results summarized in Table 4 point to several conclusions. Firstly, given that the proposed approach achieves the best results in 8 out of 10 test cases, the results suggest that it is indeed versatile enough to capture relevant information, regardless of whether that comes from the attributes of vertices, their proximity, or connectedness of vertices, even without parameter tuning. Moreover, when combined with the spectral clustering, our approach consistently obtains good results on all data sets, while the competitor approaches achieve good results if the problem fits their bias. Secondly, the results show that one has to consider not only the bias of the similarity measure, but the bias of the clustering algorithm as well, which is evident on most data sets where spectral clustering achieves substantially better performance than hierarchical clustering. Finally, ReCeNT and most of the approaches tend to be

[8] We were not able to make HS work on this data set as it assumes edges between compound vertices which are non-existing in this data set

Table 5: Performance of ReCeNT with different parameter settings. The upper part of the table presents results with the neighbourhood trees with depth of 1, whereas the bottom part contains the results with depth set to 2. The parameters in italic indicate the best performance achieved.

Parameters	**Muta**		**UWCSE**		**WebKB**		**Terror**		**IMDB**	
	Hier.	Spec	Hier.	Spec	Hier.	Spec	Hier.	Spec	Hier.	Spec
1,0,0,0,0	0.00	0.00	0.25	0.2	0.00	0.25	0.01	0.17	0.05	0.05
0,1,0,0,0	0.00	0.00	0.52	0.12	0.00	0.00	0.00	-0.01	0.0	0.00
0,0,1,0,0	0.00	0.00	0.05	0.1	0.00	0.1	0.00	0.00	0.14	0.13
0,0,0,1,0	0.30	0.30	0.02	-0.03	0.00	0.2	0.00	-0.01	0.17	0.17
0,0,0,0,1	0.24	0.25	0.17	0.07	0.00	0.02	-0.01	0.00	1.0	1.0
0.2,0.2,0.2,0.2,0.2	0.32	0.35	0.96	0.86	0.04	0.56	0.00	0.26	0.62	1.0
1,0,0,0,0	0.00	0.00	0.00	0.2	0.00	0.27	0.00	0.17	0.05	-0.05
0,1,0,0,0	0.00	0.00	0.03	0.16	0.00	0.00	0.00	-0.01	0.0	0.00
0,0,1,0,0	0.00	0.00	0.00	0.08	0.00	0.01	0.01	0.00	0.15	0.13
0,0,0,1,0	0.29	0.29	0.01	-0.03	0.02	0.2	-0.01	-0.01	0.00	0.00
0,0,0,0,1	0.00	0.27	0.03	-0.04	0.00	0.02	0.00	0.00	1.0	1.0
0.2,0.2,0.2,0.2,0.2	0.08	0.3	0.1	0.07	0.02	0.4	0.01	0.16	0.13	1.0

sensitive to the depth parameter, which is evident in the drastic difference in performance when different depths are used. This suggests that increasing depth of a neighbourhood tree consequently introduces more noise. Interestingly, while the results suggest that with ReCeNT the depth of 1 performs the best, the performance of kernel methods tend to increase with the depth parameter. These results justify the basic assumption of this approach that important information is contained in small local neighbourhoods.

4.3.2 ***(Q2) Relevance of components***

In the second experiment, we evaluate how relevant each of the five components in Equation 3 is. Table 5 summarizes the results. There are only two cases (Mutagenesis and IMDB) where using a single component (if it is the right one!) suffices to get results comparable to using all components (Table 5). This confirms that clustering relational data is difficult not only because one needs to choose the right source of similarity, but also because the similarity of relational objects may come from multiple sources, and one has to take all these into account in order to discover interesting clusters.

These results may explain why ReCeNT almost consistently outperforms all other methods in the first experiment. First, ReCeNT considers different sources of relational similarity; and second, it ensures that each source has a comparable impact (by normalizing the impact of each source and giving each an equal weight in the linear combination). This guarantees that if a component contains useful information, it is taken into account. If a component has no useful information, it adds some noise to the similarity measure, but the clustering process seems quite resilient to this. If *most* of the components are irrelevant, the noise can dominate the pattern. This is likely what happens in experiment 1 when depth 2 neighbourhood trees are used: too much irrelevant information is introduced at level two, dominating the signal at level one.

Table 6: Results obtained by AASC. The subscript indicates the depth of the neighbourhood tree.

Approach	IMDB	UWCSE	Mutagenesis	WebKB	Terror
ReCeNT_1	1.0	0.98	0.35	0.56	0.26
AASC_1	0.78	0.65	0.35	0.57	0.28
ReCeNT_2	1.0	0.07	0.3	0.4	0.16
AASC_2	0.67	0.23	0.3	0.4	0.23

4.3.3 ***(Q3) Learning weights in an unsupervised manner***

The first experiment shows that ReCeNT outperforms the competitor methods even without parameters being tuned. The second experiment shows that one typically has to consider multiple interpretations of similarity in order to obtain a useful clustering. A natural question to ask is whether the parameters could be learned from data in an unsupervised way. The possibility of tuning offers an additional flexibility to the user. If the knowledge about the right bias is available in advance, one can specify it through adjusting the parameters of the similarity measure, potentially achieving even better results than those presented in Table 4. However, tuning the weights in an automated and systematic way is a difficult task as there is no clear objective function to optimize in a purely unsupervised settings. Many clustering evaluation criteria, such as ARI, require a reference clustering which is not available during clustering itself. Other clustering quality measures do not require a reference clustering, but each of those has its own bias (Van Craenendonck and Blockeel, 2015).

An approach that might help in this direction is the *Affinity Aggregation for Spectral Clustering* (AASC) (Huang et al, 2012). This work extends spectral clustering to a multiple affinity case. The authors start from the position that similarity of objects often can be measured in multiple ways, and it is often difficult to know in advance how different similarities should be combined in order to achieve the best results. Thus, the authors introduce an approach that learns the weights that would, when clustered into the desired number of clusters, yield the highest intra-cluster similarity. That is achieved by iteratively optimizing: (1) the cluster assignment given the fixed weights, and (2) weights given a fixed cluster assignment. Thus, by treating each component in Equation 3 as a separate affinity matrix, this approach tries to learn their optimal combination.

We have tried AASC in ReCeNT, and the results are summarized in Table 6. These results lead to several conclusions. Firstly, in most cases AASC yields no substantial benefit or even hurts performance. This confirms that learning the appropriate bias (and the corresponding parameters) in an entirely unsupervised way is a difficult problem. The main exceptions are found for depth 2: here, a substantial improvement is found for UWCSE and TerroristAttack. This seems to indicate that the bad performance on depth 2 is indeed due to an overload of irrelevant information, and that AASC is able to weed out some of that. Still, the obtained results for depth 2 are not comparable to the ones obtained for depth 1. We conclude that tuning the weights in an unsupervised manner will require more sophisticated methods than the current state of the art.

4.3.4 ***(Q4) Performance in a supervised setting***

The previous experiments point out that the proposed dissimilarity measure performs well compared to the existing approaches, but finding the appropriate weights is difficult. Though

Table 7: Performance of the kNN classifier with different (dis)similarity measure and weight learning. The performance is expressed in terms of accuracy over the 10-fold cross validation.

Approach	IMDB	UWCSE	Mutagenesis	WebKB	Terrorists
HS	88.08	76.66	0.00	12.78	27.51
CC	88.08	**99.85**	60.08	61.07	38.28
HSAG	88.08	95.88	77.40	12.82	75.62
ReCeNT	**100**	**100**	**85.54**	**100**	**85.60**
RIBL	**100**	77.22	76.37	84.11	N/A
WLST	93.60	44.94	76.37	47.35	45.56
VE	**100**	98.26	70.60	49.33	30.00
V	93.80	43.61	70.42	47.35	44.39
RKOH	95.07	67.26	60.78	N/A	N/A

our focus is on clustering tasks, we can use our dissimilarity measure for classification tasks as well. The availability of labels offers a clear objective to optimize when learning the weights, and thus allows us to evaluate the appropriateness of ReCeNT for classification.

We have set up an experiment where we use a k nearest neighbours (kNN) classifier with each of the (dis)similarity measures. It consists of a 10-fold cross-validation, where within each training fold, an internal 10-fold cross-validation is used to tune the parameters of the similarity measure, and kNN with the tuned similarity measure is next used to classify the examples in the corresponding test fold.

The results of this experiment are summarized in Table 7. ReCeNT achieves the best performance on all data sets. On the IMDB data set, ReCeNT achieves perfect performance, as do RIBL and VE. On UWCSE, ReCeNT is 100% accurate; its closest competitor, CC, achieves 99.85%. From the classification viewpoint, these two data sets are easy: the classes are differentiable by one particular attribute or relation. On Mutagenesis and Terrorists, the difference is more outspoken: ReCeNT achieves around 85% accuracy, with its closest competitor (HSAG) achieving 76% or 77%. On WebKB, finally, ReCeNT and RIBL substantially outperform all the other approaches, with ReCeNT achieving 100% and RIBL 84.11%.

The remarkable performance of ReCeNT on WebKB is explained by inspecting the tuned weights. These reveal that ReCeNT's ability to jointly consider vertex identity, edge type distribution, and vertex attributes (in this case, words on webpages) are the reason why it performs so well. None of the other approaches take all three components into account, which is why they achieve substantially worse results.

These results clearly show that accounting for several views of similarity is beneficial for relational learning. Moreover, the availability of labelled information is clearly helpful and ReCeNT is capable of successfully adapting its bias towards the needs of the data set.

*4.3.5 **(Q5) Runtime comparison***

Table 8 presents a comparison of runtimes for each approach. All the experiments were run on a computer with 3.20 GHz of CPU power and 32 GB RAM. The runtimes include the construction of supporting structures (neighbourhood trees and context descriptors), calculation of similarity between all pairs of vertices, and clustering. The measured runtimes are consistent with the previously discussed complexities of the approaches. HS, HSAG, CC, ReCeNT and kernel approaches (excluding RKOH) are substantially more efficient than the

Table 8: Runtime comparison in minutes (rounded up to the closest integer). The runtimes include the construction of supporting structures and time needed to calculate a similarity between each pair of vertices in a given hypergraph. Note that graph kernel measures (in italic) are obtained using the external software provided with Sugiyama and Borgwardt (2015). N/A indicates that the calculation took more than 24 hours.

Approach	**IMDB**	**UWCSE**	**Mutagenesis**	**WebKB**	**Terror**
HS	1	1	N/A	1	1
CC_2	1	1	1	5	1
CC_4	1	1	1	8	8
$HSAG_1$	1	1	1	2	2
$HSAG_2$	1	1	1	5	2
$ReCeNT_1$	1	1	1	2	2
$ReCeNT_2$	1	1	3	10	5
$RIBL_1$	1	2	540	1320	N/A
$RIBL_2$	2	5	N/A	N/A	N/A
$WLST_{1,5}$	1	1	1	1	1
$WLST_{1,10}$	1	1	1	1	1
$WLST_{2,5}$	1	1	1	4	5
$WLST_{2,10}$	1	1	1	4	5
VE_1	1	1	1	1	2
$RKOH_{1,2}$	1	2	10	N/A	N/A
$RKOH_{1,4}$	N/A	N/A	N/A	N/A	N/A
$RKOH_{2,2}$	N/A	N/A	N/A	N/A	N/A
$RKOH_{2,4}$	N/A	N/A	N/A	N/A	N/A

remaining approaches. This is not surprising, as HS, HSAG and CC use very limited information. It is, however, interesting to see that ReCeNT and WLST, which use substantially more information, take only slightly more time to compute, while achieving substantially better performance on most data sets. These approaches are also orders of magnitude more efficient than RIBL and RKOH, which did not complete on most data sets with depth set to 2. That is particularly the case for RKOH which did not complete in 24 hours even with the depth of 1, when the walk length was set to 4.

5 Conclusion

In this work we propose a novel dissimilarity measure for clustering relational objects, based on a hypergraph interpretation of a relational data set. In contrast with the previous approaches, our approach takes multiple aspects of relational similarity into account, and allows one to focus on a specific vertex type of interest, while at the same time leveraging the information contained in other vertices. We develop the dissimilarity measure to be versatile enough to capture relevant information, regardless whether it comes from attributes, proximity or connectedness in a hyper-graph. To make our approach efficient, we introduce neighbourhood trees, a structure to compactly represent the distribution of attributes and hyperedges in the neighbourhood of a vertex. Finally, we experimentally evaluate our approach on several data sets on both clustering and classification tasks. The experiments show that the proposed method often achieves better results than the competitor methods with regards to the quality of clustering and classification, showing that it indeed is versatile enough to adapt to each data set individually. Moreover, we show that the proposed approach, though more expressive, is as efficient as the state-of-the-art approaches. One open challenge is to which extent the parameters of the proposed similarity measure can be learnt from data in

an unsupervised (or a semi-supervised) way. We conducted experiments with the *affinity aggregation* approaches that demonstrated the difficulty of this problem. The proposed similarity measure is sensitive to the depth of a neighbourhood tree, which poses a problem when large neighbourhoods have to be compared. However, the experiments demonstrated that the depth of 1 often suffices.

Future work. This work can be extended in several directions. First, there is a number of options concerning the choice of the weights of the proposed similarity measure. Learning the weights works well when class labels are available, but is difficult in an unsupervised setting. In semi-supervised classification or constraint-based clustering (Wagstaff et al, 2001), limited information is available that may help tune the weights. A small number of labels or pairwise constraints (must-link / cannot-link) may suffice to tune the weights in ReCeNT.

The second direction comes from the field of *multiple kernel learning* (Gonen and Alpaydin, 2011). The field of multiple kernel learning is concerned with finding an optimal combination of fixed kernel sets, and might be inspirational in learning the weights directly from data. In contrast to many relational clustering techniques, our approach with neighbourhood trees allows us to construct a prototype - a representative example of a cluster, which many of the clustering algorithms require. Moreover, constructing a prototype of a cluster might be of great help analysing the properties of objects clustered together. Integrating our measure into very scalable clustering methods such as BIRCH (Zhang et al, 1996), would allow one to cluster very large hypergraphs. An interesting extension would be to modify the summations over levels of neighbourhood trees into weighted sums over the same levels, following the intuition that the vertices further from the vertex of interest are less relevant, but at the same time giving them a chance to make a difference.

Acknowledgements This research is supported by Research Fund KU Leuven (GOA/13/010). The authors thank the anonymous reviewers for their helpful feedback.

References

Bader DA, Meyerhenke H, Sanders P, Wagner D (eds) (2013) Graph Partitioning and Graph Clustering - 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, USA, February 13-14, 2012. Proceedings, Contemporary Mathematics, vol 588, American Mathematical Society, DOI 10.1090/conm/588, URL http://dx.doi.org/10.1090/conm/588

Bai L, Ren P, Hancock ER (2014) A hypergraph kernel from isomorphism tests. In: Proceedings of the 2014 International Conference on Pattern Recognition, IEEE Computer Society, Washington, DC, USA, ICPR '14, pp 3880–3885

Bickel S, Scheffer T (2004) Multi-view clustering. In: Proceedings of the Fourth IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA, ICDM '04, pp 19–26

Bille P (2005) A survey on tree edit distance and related problems. Elsevier Science Publishers Ltd., Essex, UK, vol 337, pp 217–239

Camacho R, Fonseca NA, Rocha R, Costa VS (2007) ILP : - just trie it. In: Inductive Logic Programming, 17th International Conference, ILP, Corvallis, OR, USA, pp 78–87

Cook DJ, Holder LB (2006) Mining Graph Data. John Wiley & Sons

De Raedt L (2008) Logical and relational learning. Cognitive Technologies, Springer

Dzeroski S, Blockeel H (2004) Multi-relational data mining 2004: workshop report. SIGKDD Explorations 6(2):140–141, DOI 10.1145/1046456.1046481, URL http://doi.acm.org/10.1145/1046456.1046481

Emde W, Wettschereck D (1996) Relational instance based learning. In: Saitta L (ed) Proceedings 13th International Conference on Machine Learning (ICML 1996), July 3-6, 1996, Bari, Italy, Morgan-Kaufman Publishers, San Francisco, CA, USA, pp 122–130

Estivill-Castro V (2002) Why so many clustering algorithms: A position paper. SIGKDD Explor Newsl 4(1):65–75

Fonseca NA, Santos Costa V, Camacho R (2012) Conceptual clustering of multi-relational data. In: Muggleton SH, Tamaddoni-Nezhad A, Lisi FA (eds) Inductive Logic Programming: 21st International Conference, ILP 2011, Windsor Great Park, UK, July 31 – August 3, 2011, Revised Selected Papers, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 145–159

Frasconi P, Costa F, De Raedt L, De Grave K (2014) klog: A language for logical and relational learning with kernels. Artif Intell 217:117–143

Getoor L, Taskar B (2007) Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press

Gonen M, Alpaydin E (2011) Multiple kernel learning algorithms. J Mach Learn Res 12:2211–2268

Haussler D (1999) Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA

Huang HC, Chuang YY, Chen CS (2012) Affinity aggregation for spectral clustering. In: International Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, pp 773–780

Kirsten M, Wrobel S (1998) Relational distance-based clustering. In: Lecture Notes in Computer Science, Springer-Verlag, vol 1446, pp 261–270

Kok S, Domingos P (2010) Learning markov logic networks using structural motifs. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 551–558

Lovász L (1996) Random walks on graphs: A survey. In: Miklós D, Sós VT, Szőnyi T (eds) Combinatorics, Paul Erdős is Eighty, vol 2, János Bolyai Mathematical Society, Budapest, pp 353–398

Morey LC, Agresti A (1984) The measurement of classification agreement: An adjustment to the rand statistic for chance agreement. Educational and Psychological Measurement 44(1):33–37

Muggleton S, De Raedt L (1994) Inductive logic programming: Theory and methods. J Log Program 19/20:629–679, DOI 10.1016/0743-1066(94)90035-3, URL http://dx.doi.org/10.1016/0743-1066(94)90035-3

Neville J, Adler M, Jensen D (2003) Clustering relational data using attribute and link information. In: Proceedings of the Text Mining and Link Analysis Workshop, 18th International Joint Conference on Artificial Intelligence, pp 9–15

Ng AY, Jordan MI, Weiss Y (2001) On spectral clustering: Analysis and an algorithm. In: Advances in neural information proicessing systems, MIT Press, pp 849–856

Ong IM, Castro Dutra I, Page D, Costa VS (2005) Mode directed path finding. In: 16th European Conference on Machine Learning, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 673–681

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. Journal of Machine

Learning Research 12:2825–2830
Perlich C, Provost F (2006) Distribution-based aggregation for relational learning with identifier attributes. Mach Learn 62(1-2):65–105, DOI 10.1007/s10994-006-6064-1, URL http://dx.doi.org/10.1007/s10994-006-6064-1
Pfeiffer JJ III, Moreno S, La Fond T, Neville J, Gallagher B (2014) Attributed graph models: Modeling network structure with correlated attributes. In: Proceedings of the 23rd International Conference on World Wide Web, ACM, New York, NY, USA, WWW '14, pp 831–842
Rand W (1971) Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66(336):846–850
Richards BL, Mooney RJ (1992) Learning relations by pathfinding. In: Proc. of AAAI-92, San Jose, CA, pp 50–55
Sen P, Namata GM, Bilgic M, Getoor L, Gallagher B, Eliassi-Rad T (2008) Collective classification in network data. AI Magazine 29(3):93–106
Shervashidze N, Borgwardt K (2009) Fast subtree kernels on graphs. In: Proceedings of the Neural Information Processing Systems Conference NIPS 2009, Neural Information Processing Systems Foundation, pp 1660–1668
Shervashidze N, Schweitzer P, van Leeuwen EJ, Mehlhorn K, Borgwardt KM (2011) Weisfeiler-lehman graph kernels. J Mach Learn Res 12:2539–2561
Sugiyama M, Borgwardt K (2015) Halting in random walk kernels. In: Advances in Neural Information Processing Systems 28, Curran Associates, Inc., pp 1639–1647
Van Craenendonck T, Blockeel H (2015) Using internal validity measures to compare clustering algorithms. In: AutoML Workshop at 32nd International Conference on Machine Learning, Lille, 11 July 2015, pp 1–8, URL https://lirias.kuleuven.be/handle/123456789/504712
Wachman G, Khardon R (2007) Learning from interpretations: a rooted kernel for ordered hypergraphs. In: Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007, pp 943–950
Wagstaff K, Cardie C, Rogers S, Schrödl S (2001) Constrained k-means clustering with background knowledge. In: Proceedings of the Eighteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, pp 577–584
Ward JH (1963) Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association 58(301):236–244
Witsenburg T, Blockeel H (2011) Improving the accuracy of similarity measures by using link information. In: Foundations of Intelligent Systems - 19th International Symposium, ISMIS 2011, Warsaw, Poland, June 28-30, 2011. Proceedings, pp 501–512
Zhang T, Ramakrishnan R, Livny M (1996) Birch: An efficient data clustering method for very large databases. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '96, pp 103–114
Zhao H, Robles-Kelly A, Zhou J (2011) On the use of the chi-squared distance for the structured learning of graph embeddings. In: Proceedings of the 2011 International Conference on Digital Image Computing: Techniques and Applications, IEEE Computer Society, Washington, DC, USA, DICTA '11, pp 422–428