Mach Learn (2017) 106:1523–1545
DOI 10.1007/s10994-017-5644-6

An expressive dissimilarity measure for relational clustering using neighbourhood trees

Sebastijan Dumančić[1] · **Hendrik Blockeel**[1]

Received: 22 October 2016 / Accepted: 14 May 2017 / Published online: 5 June 2017

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 sets 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

Editors: Kurt Driessens, Dragi Kocev, Marko Robnik-Šikonja, and Myra Spiliopoulou.

✉ Sebastijan Dumančić
sebastijan.dumancic@cs.kuleuven.be

Hendrik Blockeel
hendrik.blockeel@cs.kuleuven.be

1 Department of Computer Science, KU Leuven, Celestijnenlaan 200A, Heverlee, Belgium

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 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 Fig. 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.

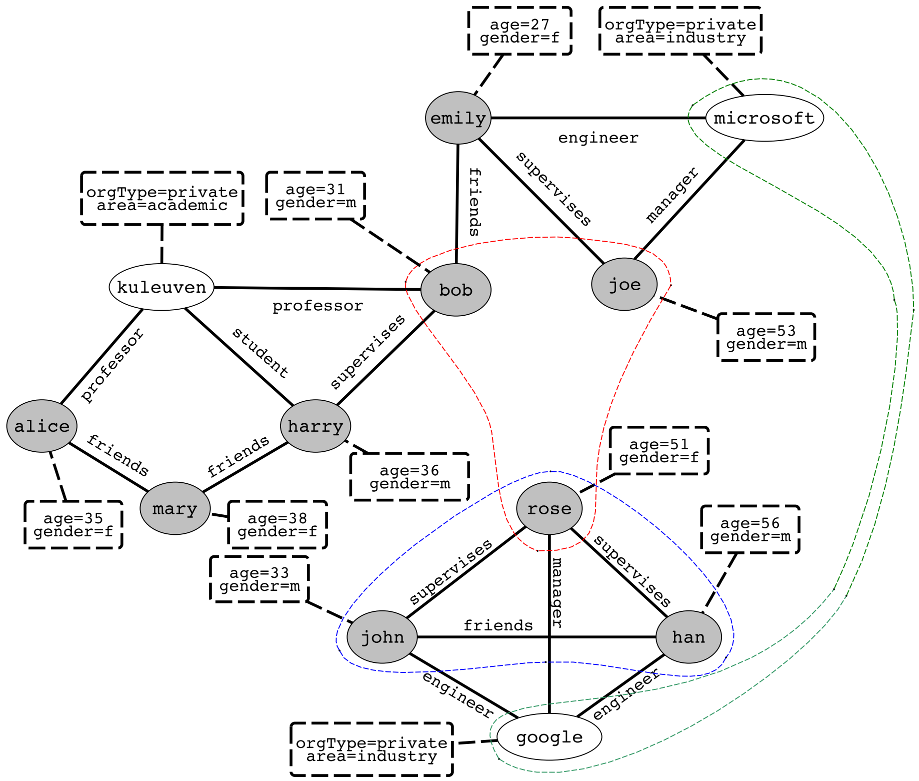


Fig. 1 An illustration of a relational data set containing people and organizations, and different clusters one might find in it. Instances—people and organizations, are represented by vertices, while relationships among them are represented with edges. The rectangles list an associated set of attributes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

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

(a)

```
person(bob,25,m,msc)
person(emily,27,f,msc)
organization(kuleuven,private,academic)
organization(microsoft,private,industry)
friends(bob,emily)
friends(emily,bob)
works_for(bob,kuleuven,professor,2000)
works_for(emily,microsoft,engineer,2300)
```

(b)

person

PName	Age	Gender	Education
bob	25	m	msc
emily	27	f	msc

organization

OName	OrgType	Area
kuleuven	private	academic
microsoft	private	industry

works_for

PName	OName	Position	Salary
bob	kuleuven	professor	2000
emily	microsoft	engineer	2300

friends

PName1	PName2
bob	emily
emily	bob

(c)

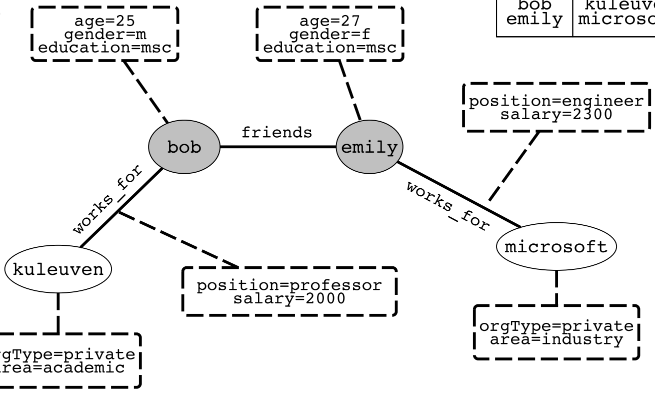


Fig. 2 Representation paradigms of relational data. **a** Represents the relational data set as a set of logical facts; the upper part represents the definition of each predicate, while the bottom part lists all facts. **b** Illustrates a *database view* of the relational data set, where each logical predicate is associated with a single database table. **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

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 entity (`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 $\tau : (V \cup E) \rightarrow T_V \cup T_E$ 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 Fig. 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 $H = (V, E, \tau, \lambda)$
         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 $d' \leq d$ 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

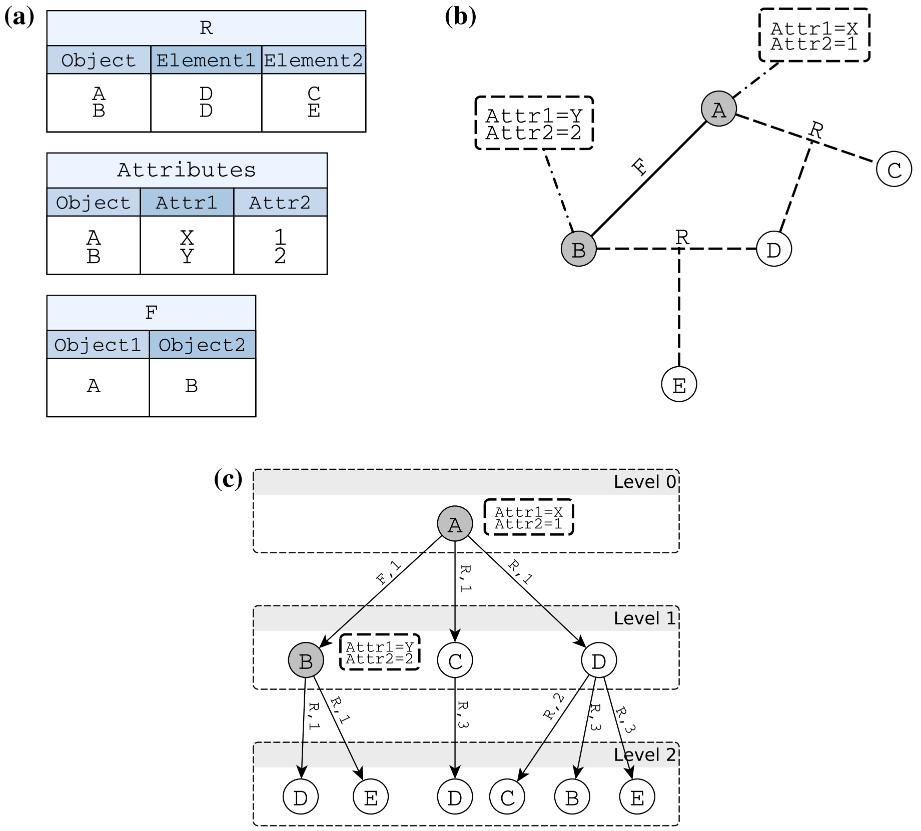
The main idea behind the proposed dissimilarity measure is to express a wide range of similarity biases that can emerge in relational data, as discussed and exemplified in Sect. 1. The proposed dissimilarity measure compares two vertices by comparing their neighbourhood trees. It does this by comparing, 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, the dissimilarity is defined as



R		
Object	Element1	Element2
A	D	C
B	D	E

Attributes		
Object	Attr1	Attr2
A	X	1
B	Y	2

F	
Object1	Object2
A	B

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. **a** Contains the database view of the domain. **b** Contains the corresponding hypergraph view. Here, edges are represented with full lines, while hyperegdges are represented with *dashed lines*. Finally, **c** Contains the corresponding *neighbourhood tree* for the vertex A

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

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)
- $V_t^l(g)$ is the multiset of vertices of type t at depth l
- $B_{t,a}^l(g)$ 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 Fig. 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*

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

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 Eq. 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 Sects. 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 Sect. 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 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

Table 1 Aspects of similarity considered by different approaches

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

✓ denotes full consideration, ⋍ partial and × no consideration at all

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 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

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	$O\left(N^2(E+A)^{2d+2l}\right)$

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.

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

Table 3 Characteristics of the data sets used in experiments

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

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

(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.

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 $w_i = 1, w_{j,j\neq i} = 0$.

[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/.

(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 h 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 as an aggregate 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), 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.

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

Table 4 Performance of all approaches on three data sets

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

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
Bold values indicate the best-obtained performance for each dataset

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, $\text{WLST}_{1,*}$ 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 $\text{WSLT}_{2,*}$ 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, $\text{WLST}_{1,*}$ 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 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.

[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

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

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

4.3.2 (Q2) Relevance of components

In the second experiment, we evaluate how relevant each of the five components in Eq. 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.

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

Table 6 Results obtained by AASC

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

The subscript indicates the depth of the neighbourhood tree

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 Eq. 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 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

Table 7 Performance of the kNN classifier with different (dis)similarity measure and weight learning

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

The performance is expressed in terms of accuracy over the 10-fold cross validation
Bold values indicate the best-obtained performance for each dataset

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 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

Table 8 Runtime comparison in minutes (rounded up to the closest integer)

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

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 h

2. That is particularly the case for RKOH which did not complete in 24 h 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, D. A., Meyerhenke, H., Sanders, P., & Wagner, D. (Eds) (2013). Graph partitioning and graph clustering. In *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

Bai, L., Ren, P., & Hancock, E. R. (2014). A hypergraph kernel from isomorphism tests. In *Proceedings of the 2014 international conference on pattern recognition, ICPR '14* (pp. 3880–3885), IEEE Computer Society, Washington, DC, USA

Bickel, S., & Scheffer, T. (2004) Multi-view clustering. In *Proceedings of the fourth IEEE international conference on data mining, ICDM '04* (pp. 19–26), IEEE Computer Society, Washington, DC, USA.

Bille, P. (2005). *A survey on tree edit distance and related problems* (Vol. 337). Essex, UK: Elsevier Science Publishers Ltd.

Camacho, R., Fonseca, N. A., Rocha, R., & Costa, V. S. (2007). ILP:-just trie it. In *17th international conference on inductive logic programming, ILP* (pp. 78–87), Corvallis, OR, USA.

Cook, D. J., & Holder, L. B. (2006). *Mining graph data*. Hoboken: John Wiley & Sons.

De Raedt, L. (2008). *Logical and relational learning. Cognitive technologies*. Berlin: Springer.

Dzeroski, S., & Blockeel, H. (2004). Multi-relational data mining 2004: Workshop report. *SIGKDD Explorations*, *6*(2), 140–141. doi:10.1145/1046456.1046481.

Emde, W., & Wettschereck, D. (1996). Relational instance based learning. In L. Saitta (Ed.), *Proceedings 13th international conference on machine learning (ICML 1996)* (pp. 122–130), July 3–6, 1996. USA: Bari, Italy, Morgan-Kaufman Publishers, San Francisco, CA.

Estivill-Castro, V. (2002). Why so many clustering algorithms: A position paper. *SIGKDD Explorations Newsletter*, *4*(1), 65–75.

Fonseca, N. A., Santos Costa, V., & Camacho, R. (2012). Conceptual clustering of multi-relational data. In S. H. Muggleton, A. Tamaddoni-Nezhad, & F. A. Lisi (Eds.), *Inductive logic programming: 21st international conference, ILP 2011* (pp. 145–159), Windsor Great Park, UK, July 31–August 3, 2011. Revised Selected Papers. Berlin: Springer.

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

Getoor, L., & Taskar, B. (2007). *Introduction to statistical relational learning (adaptive computation and machine learning)*. Cambridge: The MIT Press.

Gonen, M., & Alpaydin, E. (2011). Multiple kernel learning algorithms. *Journal of Machine Learning Research*, *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, H. C., Chuang, Y. Y., & Chen, C. S. (2012) Affinity aggregation for spectral clustering. In *International conference on computer vision and pattern recognition* (pp. 773–780), IEEE Computer Society.

Kirsten, M., & Wrobel, S. (1998). Relational distance-based clustering. In *Lecture notes in computer science* (Vol. 1446, pp. 261–270). Springer-Verlag.

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 D. Miklós, V. T. Sós, & T. Szőnyi (Eds.), *Combinatorics, Paul Erdős is eighty* (Vol. 2, pp. 353–398). Budapest: János Bolyai Mathematical Society.

Morey, L. C., & 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. *The Journal of Logic Programming*, *19*(20), 629–679. doi: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, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (pp. 849–856). MIT Press.

Ong, I. M., Castro Dutra, I., Page, D., & Costa, V. S. (2005). Mode directed path finding. In *16th European conference on machine learning* (pp. 673–681). Berlin: Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (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. *Machine Learning*, *62*(1–2), 65–105. doi:10.1007/s10994-006-6064-1.

Pfeiffer, J. J. 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, WWW '14* (pp. 831–842), ACM, New York, NY, USA.

Rand, W. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, *66*(336), 846–850.

Richards, B. L., & Mooney, R. J. (1992). Learning relations by pathfinding. In *Proceedings of of AAAI-92* (pp 50–55), San Jose, CA.

Sen, P., Namata, G. M., 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* (pp. 1660–1668), Neural Information Processing Systems Foundation.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., & Borgwardt, K. M. (2011). Weisfeiler–Lehman graph kernels. *Journal of Machine Learning Research*, *12*, 2539–2561.

Sugiyama, M., & Borgwardt, K. (2015). Halting in random walk kernels. In *Advances in neural information processing systems 28* (pp 1639–1647). Curran Associates, Inc.

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, July 11, 2015, (pp 1–8) https://lirias.kuleuven.be/handle/123456789/504712

Wachman, G., & Khardon, R. (2007). Learning from interpretations: a rooted kernel for ordered hypergraphs. In *Proceedings of the twenty-fourth international conference on machine learning (ICML 2007)* (pp. 943–950), Corvallis, Oregon, USA, June 20–24, 2007.

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, ICML '01* (pp. 577–584). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ward, J. H. (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—Proceedings of 19th international symposium, ISMIS 2011* (pp. 501–512), Warsaw, Poland, June 28–30, 2011.

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, SIGMOD '96* (pp. 103–114), ACM, New York, NY, USA

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, DICTA '11* (pp. 422–428), IEEE Computer Society, Washington, DC, USA.