



**HAL**  
open science

## Measuring structural similarity between RDF graphs

Pierre Maillot, Carlos Bobed

► **To cite this version:**

Pierre Maillot, Carlos Bobed. Measuring structural similarity between RDF graphs. 33rd Annual ACM Symposium on Applied Computing, Apr 2018, Pau, France. pp.1960-1967, 10.1145/3167132.3167342 . hal-01940449

**HAL Id: hal-01940449**

**<https://hal.science/hal-01940449>**

Submitted on 24 Jan 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Measuring Structural Similarity Between RDF Graphs

Pierre Maillot  
Univ Rennes, CNRS, IRISA  
F-35000, Rennes, France  
pierre.maillot@irisa.fr

Carlos Bobed  
Univ Rennes, CNRS, IRISA  
F-35000, Rennes, France  
carlos.bobed-lisbona@irisa.fr

## ABSTRACT

In the latest years, there has been a huge effort to deploy large amounts of data, making it available in the form of RDF data thanks, among others, to the Linked Data initiative. In this context, using shared ontologies has been crucial to gain interoperability, and to be able to integrate and exploit third party datasets. However, using the same ontology does not suffice to successfully query or integrate external data within your own dataset: the actual usage of the vocabulary (e.g., which concepts have instances, which properties are actually populated and how, etc.) is crucial for these tasks. Being able to compare different RDF graphs at the actual usage level would indeed help in such situations. Unfortunately, the complexity of graph comparison is an obstacle to the scalability of many approaches.

In this article, we present our *structural similarity measure*, designed to compare structural similarity of low-level data between two different RDF graphs according to the patterns they share. To obtain such patterns, we leverage a data mining method (KRIMP) which allows to extract the most descriptive patterns appearing in a transactional database. We adapt this method to the particularities of RDF data, proposing two different conversions for an RDF graph. Once we have the descriptive patterns, we evaluate how much two graphs can compress each other to give a numerical measure depending on the common data structures they share. We have carried out several experiments to show its ability to capture the structural differences of actual vocabulary usage.

## CCS CONCEPTS

•Information systems → Similarity measures; Graph-based database models; Data mining; Content analysis and feature selection; •Computing methodologies → Knowledge representation and reasoning;

## KEYWORDS

Similarity; Semantic Web; Linked Data; Data Mining

### ACM Reference format:

Pierre Maillot and Carlos Bobed. 2016. Measuring Structural Similarity Between RDF Graphs. In *Proceedings of ACM SAC Conference, Pau, France, April 9-13, 2018 (SAC'18)*, 8 pages.

DOI: <https://doi.org/10.1145/3167132.3167342>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

SAC'18, Pau, France

© 2018 ACM. 978-1-4503-5191-1/18/04...\$15.00

DOI: <https://doi.org/10.1145/3167132.3167342>

## 1 INTRODUCTION

In the last years, the amount of data available in RDF kept growing both in size and diversity. Different initiatives, such as the Linked Open Data, and the progressive adoption of semantic annotations within web pages [7] are fueling this growth making available domain specific datasets which are (re)using the same ontologies. Such use of a common vocabulary and the interlinking between those datasets are the direct application of the Linked Data principles. This way of publishing data greatly increases their interoperability, and enable a number of new applications based on the integration and exploitation of new RDF data. However, sharing a common vocabulary by itself helps, but does not guarantee smooth data integration and interoperability. Two datasets using the same ontologies as background schema can use their common vocabulary in different ways. For example, instances may be distributed differently among the class hierarchy (i.e., the used part of the ontologies could be disjoint), or even when using the same concepts, the properties of the instances in each dataset may be differently populated. Therefore, when confronted to a new dataset, it is necessary to be able to check if its usage of the vocabulary (i.e., the actual structure of the data) is similar to the other datasets used in your application.

In the case of RDF data, this checking involves the comparison of two RDF graphs, which is in turn closely related to the problem of the evaluation of the distance between two graphs. Such distance has been extensively studied for generic graphs concluding that *graph edit distances* are very expensive and does not scale well, as the survey by Gao et al. [2] shows. Moreover, as RDF graphs can be seen as multi-dimensional graphs with semantics attached, it is difficult to devise an adapted edit cost function. Thus, the particularities of RDF graphs and the recent explosion in the quantity of low-level RDF data accessible pose new challenges for measures of similarity between knowledge bases.

In this paper, we propose a similarity measure based on the detection of common structural regularities between two RDF graphs. This measure, which we have called the *structural similarity measure*, is asymmetrical, grows monotonically with the number of differences between the datasets, and can detect structural inclusion (i.e., if a dataset has the same structure as a part of the other). Our approach leverages a data mining approach called KRIMP [18] to extract a set of important and descriptive patterns from RDF graphs. With this set of patterns, the KRIMP approach makes it possible to calculate the compressed size of a database. Our approach adapts this feature of KRIMP to compute a similarity measure which compares two RDF graphs according to the frequent patterns they share. As KRIMP uses transactional databases as input, we propose two different conversions from RDF graph to transactional database. Such conversions provide different levels of structural information, depending on the presence of properties and types in the neighborhood of instances in an RDF graph. To derive our measure, we revisit and reinterpret the

measure proposed by Vreeken et al. [17], where they just focused on the general similarity of the extensions of databases where the schema is completely fixed. We tested our approach using both conversions in three different experiments using the Scholarly Data dataset [19] showing the potential of our proposal.

The rest of paper is as follows. In Section 2, we briefly describe the KRIMP algorithm and some particular aspects especially relevant for understanding our proposal. Section 3 presents both the RDF graph conversions we propose, and the *structural similarity measure* calculated out from the compression ratios obtained with KRIMP. Then, in Section 4, we present the experimentation we have carried out to show different properties of our proposed measure. Finally, Section 5 discusses the related work, and we present our conclusions and future work in Section 6.

## 2 PRELIMINARIES: FREQUENT ITEMSET MINING AND THE KRIMP ALGORITHM





In data mining, the pattern mining field aims at extracting interesting subsets of a database. For this purpose, one of the most used instances of this problem is frequent pattern mining, whose goal is to extract all interesting subsets of a given database. Formally, let  $\mathcal{I}$  a set of items. A database  $\mathcal{D}$  in pattern mining is a set of *transactions* such as each transaction  $t$  is a non-empty subset of  $\mathcal{I}$ . The interesting subsets of  $\mathcal{D}$  are *itemsets*, which are defined as such  $X$  such as there is  $X \subset \mathcal{I}$  and  $X \neq \emptyset$ . In frequent pattern mining, the “interestingness” of an itemset  $X$  depends on its *support* among the transactions, i.e. the number of transactions that contain  $X$ .

$$\text{supp}_{\mathcal{D}}(X) = |\{t \in \mathcal{D} | X \subseteq t\}| \quad (1)$$

The itemsets with a support above the *minimal support* threshold *minsup* are called *frequent itemsets*. A major problem in frequent pattern mining is the very high number of frequent itemsets extracted, which is usually reduced by imposing additional constraints on the select itemsets. A common approach is to limit the search of frequent itemsets to the *closed frequent itemsets*, which are defined as such itemsets  $X$  such as there is no other itemset  $Y$  which holds  $X \subset Y$  and  $\text{supp}_{\mathcal{D}}(X) = \text{supp}_{\mathcal{D}}(Y)$ .

However, imposing constraints on itemsets is usually not enough to obtain a manageable amount of them, so further criteria are used to limit the explosion of possible patterns. In this regard, the KRIMP algorithm [18] is a pattern mining approach based on frequent pattern mining aiming at selecting the most descriptive patterns of the database by adopting the Minimal Description Length principle [14]. Citing the authors, the idea of KRIMP is to find “the set of frequent itemsets that yield the best lossless compression of the database”. One of the advantages of this approach is that there is no parameters needed. However, as finding the optimal set is known to be NP-complete, KRIMP algorithm is based on greedy heuristics.

Thus, the goal of the KRIMP algorithm is to find a *code table* giving an optimal description of a database. Such code table is a two column translation table associating itemsets with *codes* (see Figure 1). The set of itemsets contained in the code table is its *coding set*, and it contains the most describing patterns for a database  $\mathcal{D}$ . Besides, for completeness, the coding set always contains the singleton itemsets containing each individual item. In particular, the code table containing only the singleton itemsets of database  $\mathcal{D}$  is called its *standard code table*. It gives the simplest set of patterns representing a database.

Code table $CT$	
Itemset	Code
A B C	
A B	
A	
B	
C	—

**Figure 1: Example of code table in a database with  $\mathcal{I} = \{A, B, C\}$ , extracted from [18]. The gray scale indicates different codes, while the length of the boxes is related to their length.**

To encode a transaction  $t$  with a code table  $CT$ , KRIMP uses the *cover function*,  $\text{cover}(CT, t)$ , which assigns a set of disjoint itemsets from the coding set  $CS$  to  $t$ . Such set of itemsets is called the *cover* of the transaction, and is a set of non-overlapping itemsets which hold that:

$$t = \bigcup_{X \in \text{cover}(CT, t)} X \quad (2)$$

Thus, the encoding of a database is done by replacing each transaction by the codes of the itemsets of their cover. To guarantee an optimal code size, Vreeken et al. [18] gives a code length measure in bits based on the notion of *usage*. The usage of a code  $X$  from the coding table  $CT$  of a database  $\mathcal{D}$  is defined as:

$$\text{usage}_{\mathcal{D}}(X) = |\{t \in \mathcal{D} | X \in \text{cover}(CT, t)\}| \quad (3)$$

From this definition, the length of the code  $\text{code}_{CT}(X)$  of  $X$ ,  $X \in CT$  is defined as:

$$L(\text{code}_{CT}(X)) = -\log\left(\frac{\text{usage}_{\mathcal{D}}(X)}{\sum_{Y \in CT} \text{usage}_{\mathcal{D}}(Y)}\right) \quad (4)$$

Hence, the encoded length of a transaction is defined as:

$$L(t|CT) = \sum_{X \in \text{cover}(CT, t)} L(\text{code}_{CT}(X)) \quad (5)$$

The encoded size of a database  $\mathcal{D}$  is defined as:

$$L(\mathcal{D}|CT) = \sum_{t \in \mathcal{D}} L(t|CT) \quad (6)$$

The size of a code table  $CT$  of database  $\mathcal{D}$ , with a standard code table  $ST$ , is defined as:

$$L(CT|\mathcal{D}) = \sum_{X \in CT: \text{usage}_{\mathcal{D}}(X) \neq 0} L(\text{code}_{ST}(X)) + L(\text{code}_{CT}(X)) \quad (7)$$

The total encoded size of a database  $\mathcal{D}$  and its code table  $CT$  is defined as:

$$L(\mathcal{D}, CT) = L(\mathcal{D}|CT) + L(CT|\mathcal{D}) \quad (8)$$

The encoded size of a base can be seen as a reduction of the size of the base according to its regularities. Hence the ratio between the encoded size of a database according to its code table and its encoded size according to the standard code table gives a measure of its the regularity. In Section 4, we will call this ratio the *KRIMP compression ratio* defined as:

$$\text{ratio}_{KRIMP} = \frac{L(\mathcal{D}, CT)}{L(\mathcal{D}, ST)} \quad (9)$$

The closer this ratio is close to 0, the more regular the base is. On the other hand, it gets closer to 1 if the base contain few regularities.

The KRIMP algorithm takes as input a set of candidate itemsets and a set of transactions (i.e., the database), and uses an heuristic based on the size, usage and support of each the candidate itemsets to find a code table minimizing the encoded size of the transactions in a best effort manner. Usually the candidate itemsets are frequent itemsets extracted by another algorithm from the set of transactions, such as in our experiments in Section 4 where we used all closed frequent itemsets extracted with FPClose [4]. The code table obtained by the KRIMP algorithm can be seen as an ordered set of descriptive itemsets, according to the KRIMP heuristic, which can be used to compress the set of transactions.

### 3 MEASURING SIMILARITIES BETWEEN RDF GRAPHS

Depending on the source of the data (e.g., data publishers, human contributors, or automated data extractors), the contents of a knowledge base can be described in different ways even when using the same vocabulary. This can happen due to many factors, such as the availability of data for populating the properties, the focus of the subject of the base, or the modeling skills and habits of the people responsible for making the data available.

Differences in the way the vocabularies are used are a problem when it comes to integrating data from different sources. We propose an approach to detect them by taking into account the pattern sharing degree derived from the exploitation of KRIMP code tables. To achieve this, we first propose a way to convert RDF data into transactional data which keeps the structural elements of an instance description that can be used in patterns; and then, we move onto our proposed similarity measure.

#### 3.1 From RDF to Transactions

As seen in Section 2, frequent itemsets mining approaches are applied over sets of transactions. Thus, to be able to apply KRIMP on RDF data, our approach needs to convert the graph-based RDF data to that kind of input. Such a conversion must provide a description of the instances in the RDF graph that can be useful to detect structural patterns. For this purpose, we propose to transform the description of individuals to sets of items indicating the apparition of different RDF elements in their neighborhood (e.g., RDF resources or structures).

In particular, we propose two conversions depending on the amount of information included in the transactions, namely *property-based*, and *property-class-based* conversions (PB and PCB from now on, respectively). We build such conversions on the notion of *description* of a resource, which is defined as follows:

*Definition 3.1.* Let be  $r$  a resource included in an RDF graph  $\mathcal{B}$ , we define its description  $desc(r)$  as

$$desc(r) = \{(a, R, b) \mid (a, R, b) \in \mathcal{B} \wedge (a=r \vee b=r)\} \quad (10)$$

In a transaction associated to an instance, PB conversion includes items representing the asserted types of the instance, and the usage of properties in the ingoing and outgoing triples of the instance neighborhood. Thus, the items used in PB conversion depend on the classes and properties used in the vocabulary of an RDF graph, and are defined as follows:

*Definition 3.2.* Let  $\mathcal{O}$  be the vocabulary of a given RDF graph  $\mathcal{B}$ . Let  $N_C$ ,  $N_R$ , and  $N_D$  the concept, role, and data property names used

in  $\mathcal{O}$ . Then, in PB conversion, we define its *itemset vocabulary*  $\mathcal{I}_{PB}$  as

$$\mathcal{I}_{PB} = \mathcal{I}_T \cup \mathcal{I}_{P_{IN}} \cup \mathcal{I}_{P_{OUT}} \quad (11)$$

with

$$\begin{aligned} \mathcal{I}_T &= \{i_c \mid c \in N_C\} \\ \mathcal{I}_{P_{IN}} &= \{i_{p_{in}} \mid p \in N_R\} \\ \mathcal{I}_{P_{OUT}} &= \{i_{p_{out}} \mid p \in (N_R \cup N_D)\} \end{aligned}$$

Then, we define the transaction associated to an instance according to the PB conversion as follows:

*Definition 3.3.* The PB conversion of a resource  $r$  in an RDF graph  $\mathcal{B}$  is the transaction  $t_{r_{PB}}$  generated by the application of the following PB-conversion rules until no new item is added to  $t_{r_{PB}}$ :

- (a)  $\exists (r, \text{rdf:type}, c) \in desc(r) \wedge c \in N_C \implies i_c \in t_{r_{PB}}$
- (b)  $\exists (r, p, r') \in desc(r) \wedge p \in (N_R \cup N_D) \implies i_{p_{out}} \in t_{r_{PB}}$
- (c)  $\exists (r', p, r) \in desc(r) \wedge p \in N_R \implies i_{p_{in}} \in t_{r_{PB}}$

The PCB conversion extends the PB conversion by adding items which represent combinations of the properties and the types of the other resources that a resource is linked to. In particular, PCB conversion extends the list of items used in PB conversion as follows:

*Definition 3.4.* Let  $\mathcal{O}$  be the vocabulary of a given RDF graph  $\mathcal{B}$ . Let  $N_C$ ,  $N_R$ , and  $N_D$  the concept, role, and data property names used in  $\mathcal{O}$ . Then, in PCB conversion, we define its *itemset vocabulary*  $\mathcal{I}_{PCB}$  as

$$\mathcal{I}_{PCB} = \mathcal{I}_{PB} \cup \mathcal{I}_{T, P_{IN}} \cup \mathcal{I}_{T, P_{OUT}} \quad (12)$$

with  $\mathcal{I}_{PB}$  as defined in Definition 3.3 and

$$\begin{aligned} \mathcal{I}_{T, P_{IN}} &= \{i_{c, p_{in}} \mid (c, p) \in N_C \times N_R\} \\ \mathcal{I}_{T, P_{OUT}} &= \{i_{c, p_{out}} \mid (c, p) \in N_C \times (N_R \cup N_D)\} \end{aligned}$$

Obviously,  $\mathcal{I}_T$ ,  $\mathcal{I}_{P_{IN}}$ ,  $\mathcal{I}_{P_{OUT}}$ ,  $\mathcal{I}_{T, P_{IN}}$  and  $\mathcal{I}_{T, P_{OUT}}$  are forced to be disjoint. Again, we define the transaction associated to an instance according to the PCB conversion as follows:

*Definition 3.5.* The PCB conversion of a resource  $r$  in a graph  $\mathcal{B}$  is the transaction  $t_{r_{PCB}}$  generated by the application of the PB-conversion rules along with the following PCB-conversion rules until no new item is added to  $t_{r_{PCB}}$ :

- (d)  $\exists \{(r, p, r'), (r', \text{rdf:type}, c)\} \subset desc(r) \wedge c \in N_C \wedge p \in (N_R \cup N_D) \implies i_{c, p_{out}} \in t_{r_{PCB}}$
- (e)  $\exists \{(r', p, r), (r', \text{rdf:type}, c)\} \subset desc(r) \wedge c \in N_C \wedge p \in N_R \implies i_{c, p_{in}} \in t_{r_{PCB}}$

PB and PCB conversions transcribe the distribution of types and properties in the neighborhood of the instances of an RDF graph. In a sense, PB conversion is similar to the notion of *individual type* proposed by Glimm et al. [3]; however, our conversion also reflects the data distribution. PCB conversion extends such description with a description of the actual interconnection between instances of different types.

We acknowledge that other conversions could be applied, providing further structural information to the pattern mining algorithm. However, we have to take into account also that usually the data mining algorithms increase their cost mainly with the number of possible items, so we have focused on proposing and studying the usefulness of these conversions. PB and PCB transcribe important structural features for the description of instances while limiting the number of items used.

### 3.2 Measuring Similarity Through Compression

As above mentioned, the KRIMP algorithm extracts the most descriptive patterns from a database of transactions. Besides, its compression capabilities allows to measure the regularity of such database as we have introduced in Section 2. We propose to exploit such compression capabilities (i.e., compute the encoded size depending on the patterns appearing in the data) to propose a measure of similarity between the data structure of two RDF graphs. In particular, our measure is defined as:

*Definition 3.6.* Let  $\mathcal{B}_1$  and  $\mathcal{B}_2$  be two given RDF bases, and  $\mathcal{D}_1$  and  $\mathcal{D}_2$  their respective conversions into transaction databases. The *structural similarity measure* to compare  $\mathcal{B}_1$  to  $\mathcal{B}_2$ , regardless the particular conversion applied, is defined as:

$$\text{sim}(\mathcal{B}_1|\mathcal{B}_2) = \frac{L(\mathcal{D}_1|CT_{\mathcal{D}_2})}{L(\mathcal{D}_1|CT_{\mathcal{D}_1})} \quad (13)$$

In other words, when comparing  $\mathcal{B}_1$  against  $\mathcal{B}_2$ , our measure compares how well the code table of  $\mathcal{D}_2$  is able to compress the transactions in  $\mathcal{D}_1$  regarding the compression achieved by the code table of  $\mathcal{D}_1$ . It is a variation of the measure proposed by Vreeken et al. [17] to compare database extensions, adapted to our needs (see Section 5 for further discussion).

When comparing one RDF graph according to the other, our *structural similarity measure* indicates that a graph contains frequent data structures that also appear in the other graph. The main idea is that the more RDF graphs share patterns, the structurally closer they are, as KRIMP has selected such patterns independently for both of them. The comparison is done calculating the ratio between the encoded sizes of  $\mathcal{D}_1$  using such both code tables. Note that it can be asymmetrical due to the inclusion of one code table into another (i.e., one graph may contain most of the other’s patterns but not the opposite). Besides, when the datasets use different vocabularies  $V_1$  and  $V_2$ , the actual used vocabulary is  $V = V_1 \cup V_2$  as, when compressing the graphs, we update the usages of the each code tables to get the fairest code length for both.

As our measure is a ratio, close structural similarity appears as its value gets close to 1. However, we have to remark that, as KRIMP applies a heuristic, it might not find the optimal code table and we can witness a structural similarity measure below 1. Such values indicate important structural similarity to a point where descriptive patterns for the evaluated RDF graph appear in the reference code table but not in its own code table.

Our measure has the advantage that it only needs to compute the code table and the transaction conversion once for any given RDF graph to enable it to be compared to any other RDF graph. In its current state, we need to make available the conversion index from RDF element to item to be able to share an RDF graph code table. However, it could be possible in future works to include code table in RDF format as meta-data associated to an RDF graph, for example using a notation inspired by the Shapes from the SHACL W3C recommendation [5].

## 4 EXPERIMENTAL EVALUATION

In this section, we present the evaluation we have carried out to evaluate how our approach captures of differences between datasets.

First, we present the experimental settings and the details of the prototype implemented; then, we move onto the different experimental results and their analysis.

### 4.1 Experimental Settings

We have gathered eleven RDF datasets describing different conferences related to the Semantic Web from the ScholarlyData [19] dataset, a refactoring of the Semantic Web Dog Food (SWDF) initiative. These datasets have been selected taking into account their size in terms of triples (see Table 1) to check the independence of our measure from the size of the datasets being compared. Besides, we specially selected two of them because they are datasets published after the refactoring from SWDF to ScholarlyData. This was done to check whether our approach could detect a change in the way that the data has been published despite using the same vocabulary. The set of eleven datasets will be referred to as the *Real Collection* from now on.

**Table 1: Size of the scholar datasets used in the experiments (in increasing order of RDF triples).**

Conference	#RDF triples	#Transactions
EKAW’16 <sup>✓*</sup>	11,278	1,866
ISWC’12	12,211	2,694
ISWC’10	12,809	2,436
ESWC’12*	13,147	2,314
ESWC’15	13,826	2,533
WWW’11*	16,689	3,137
ISWC’13*	21,128	4,679
ESWC’14	21,509	3,851
ESWC’17 <sup>✓</sup>	23,053	4,021
LREC’08	38,248	7,304
WWW’12*	39,650	7,076

<sup>✓</sup> Dataset released after the Semantic Web Dog Food refactoring.

\* Dataset included in the *Evolution Collection*.

Taking the *Real Collection* as starting point, we have built some derived collections to test how our measure values varies with the number of differences, and if it can detect structurally included datasets. To test these two points, we have used two collections, named the *Evolution* and *Inclusion Collection*, respectively.

The *Evolution Collection* contained modified versions of a subset of the *Real Collection* (see Table 1). The modifications applied were: 1) randomly delete a class assertion of a randomly selected resource (i.e., removing a (a, rdf:type, T) triple); 2) randomly delete a property assertion of a randomly selected resource (i.e., removing a (a, R, b) triple); 3) randomly modify the value of a property for a randomly selected resource (i.e., modifying  $b$  in an existing (a, R, b) triple). The new value was randomly selected from the values already present in the RDF graph for that property. Using such modifications, we built three groups of synthetic datasets: a) applying only class assertion deletions, b) applying only property deletions and modifications, and c) applying the three types of modifications. For each original dataset, each group contained three versions of five modified containing an increasing number of triples up to 50%, in 10% steps, leading to 45 derived datasets for each original one.

The *Inclusion Collection* was derived by selecting the smallest and the biggest datasets (resp. EKAW’16, WWW’12) of the *Real Collection*. We also selected the most similar dataset to the others (i.e.,

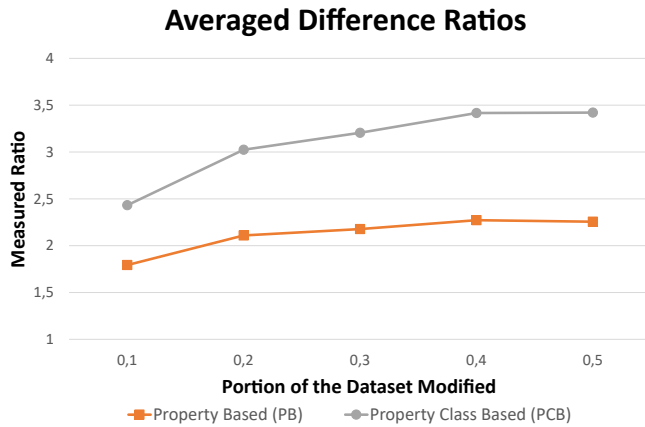


Figure 2: Averaged values of the differences measured by our approach with the two different proposed conversions.

ISWC'12) according to our measure in our first experiments (see Figure 4). We merged each of the selected datasets into the other datasets from Real Collection. For each of the selected datasets, we obtained ten datasets including them.

We have implemented our measure using Java 1.8, and used Jena 3.1.1 to handle the RDF data. We want to remark that we have focused our experiments on the ability of our approach to capture the structural differences between datasets, as the scalability of the KRIMP algorithm has been already thoroughly analyzed and shown by Vreeken et al. [18]. We note that the complexity of the KRIMP algorithm is function of the number of transactions and the number of candidates itemsets given. In our experiments, we used as candidate itemsets all closed frequent itemsets. Finally, we want to remark that we focus on graphs large enough to be used with data mining techniques where other approaches do not scale well.

## 4.2 Capturing Differences

Our first experiments aimed at evaluating the behavior of our measure during the comparison of increasingly different versions of the same dataset, using the *Evolution Collection*. Figure 2 and Figure 3 show the averaged results of the measure for the whole *Evolution Collection*, and the detail for ISWC'13 (averaged over its 45 derived datasets).

We can see how the structural similarity measure increases along with the percentage of modified triples. However, in Figure 2, we can see that when it reaches the 50%, it begins to stall. The explanation for these values can be found in the combination and nature of the modifications and the conversion. In our datasets, we interpret the decrease in the growth of our measure as the sign that our modifications changed the repartition of items less significantly than before, and thus did not make appear new patterns. When the dataset is smaller, this behavior appeared earlier<sup>1</sup>. Regarding the conversion employed, we can see how the addition of more detailed structural elements makes our approach detect more clearly the differences between the datasets. This allows us to be able to put a trade-off in the conversion we want to use: The more important it is for a given

<sup>1</sup>The rest of detailed graphs can be found at <http://sid.cps.unizar.es/projects/simStruct/>

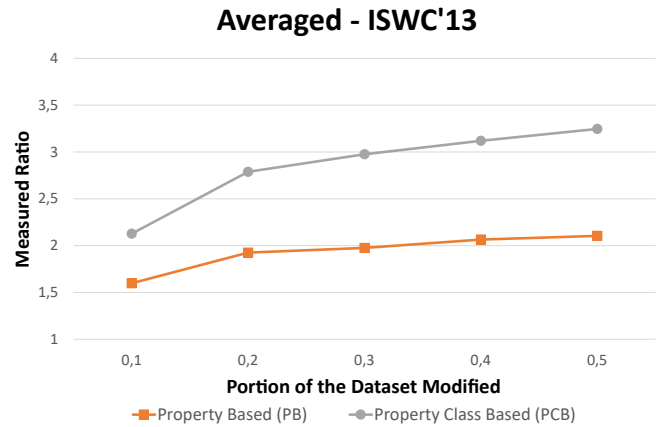


Figure 3: Averaged values of the differences measured by our approach with the two different proposed conversion for the ISWC'13 dataset.

application to capture fine-grained details, the more information should be added in the conversion step; however, this comes at the cost of adding more potential codes to the KRIMP algorithm.

This evolution scenario is very usual in semantic annotations (e.g., using JSON-LD) deployment and Linked Data endpoints: we can have an evolving dataset and, using our approach, we can assess the structural differences between versions. Besides, in these experiments we made sure that our modifications do not use new concepts or properties, which is the most challenging scenario for our approach. If we considered new concepts or properties in the compared dataset, the difference will be directly increased as by no means the initial code table can codify them.

## 4.3 Measuring Real Datasets

Our next experiments aimed at observing the behavior of our measure in a real setting by comparing the 11 datasets in the *Real Collection* against each other.

In Figure 4, we can see the results for the cross comparison for both conversions. The ScholarlyData datasets have been arranged in alphabetical and increasing year order, except for the two conferences that were released after the refactoring from SWDF (i.e., EKAW'16 and ESWC'17). The leftmost column displays the dataset against which the dataset in the column has been compared. The diagonal shows the KRIMP compression ratio of the datasets (see Definition 9); the compared datasets are really compact, meaning that their code tables contain patterns that can cover most of their data structures.

As expected, the usage of a finer-detail conversion (PCB) increases the differences. At first sight, we can see how the difference between the two newest datasets (first two rows) and the rest clearly stands out, especially when comparing the old ones against the new ones. When comparing ESWC'17 against EKAW'16, the difference, while asymmetric, is really close. When we compare the new datasets against the old ones (first two columns), we can see how the differences are lower, meaning that the code tables of the older datasets compress part of the data really well. This is explained by the difference of importance that some particular concepts and properties have gained in the new data deployment: the importance of some

Property Based (PB)	EKAW'16	ESWC'17	ESWC'12	ESWC'14	ESWC'15	ISWC'10	ISWC'12	ISWC'13	LREC'08	WWW'11	WWW'12
EKAW'16	0.103	1.216	3.612	4.824	4.457	3.175	5.294	4.087	4.762	3.992	3.312
ESWC'17	3.283	0.095	5.427	5.144	5.059	5.449	4.697	4.178	6.358	6.047	5.111
ESWC'12	3.355	3.423	0.100	1.785	1.553	3.212	2.836	2.735	2.625	4.334	1.598
ESWC'14	3.270	3.281	2.238	0.104	1.413	2.934	2.815	2.637	2.638	3.364	2.177
ESWC'15	3.184	3.115	2.269	1.792	0.101	3.110	2.869	2.756	2.638	3.647	2.230
ISWC'10	2.922	2.887	1.060	1.684	1.561	0.118	2.760	2.604	2.625	1.113	1.427
ISWC'12	1.259	1.211	0.985	1.105	1.217	0.970	0.130	1.045	1.013	0.983	0.959
ISWC'13	1.380	1.314	1.207	1.289	1.459	1.132	1.031	0.141	1.011	1.041	1.096
LREC'08	2.170	1.936	2.120	1.943	1.874	2.101	2.415	2.178	0.094	1.991	1.850
WWW'11	2.163	2.174	1.649	2.689	1.939	1.701	1.610	1.842	1.041	0.100	1.678
WWW'12	1.486	1.653	1.001	1.782	1.474	1.503	1.822	1.807	1.603	2.213	0.111
Property Class Based (PCB)	EKAW'16	ESWC'17	ESWC'12	ESWC'14	ESWC'15	ISWC'10	ISWC'12	ISWC'13	LREC'08	WWW'11	WWW'12
EKAW'16	0.059	1.555	6.845	7.977	7.285	5.986	8.918	6.493	10.843	6.301	6.177
ESWC'17	5.107	0.052	8.835	8.314	8.002	8.950	7.645	6.508	10.844	10.089	8.233
ESWC'12	5.916	6.772	0.059	2.299	2.040	4.641	4.195	3.789	4.163	6.298	2.035
ESWC'14	5.792	6.638	2.906	0.061	1.524	4.291	4.183	3.695	4.180	4.954	2.990
ESWC'15	5.675	6.423	2.967	2.236	0.062	4.487	4.237	3.816	4.179	5.257	3.051
ISWC'10	5.431	6.158	1.105	2.385	2.201	0.068	4.077	3.620	4.163	1.218	1.822
ISWC'12	2.236	2.818	0.986	1.295	1.606	0.977	0.073	1.110	1.025	0.998	1.045
ISWC'13	2.369	2.934	1.214	1.484	1.857	1.145	1.038	0.080	1.014	1.060	1.184
LREC'08	2.626	2.249	2.683	2.401	2.503	2.650	3.388	2.950	0.052	2.323	2.269
WWW'11	3.517	4.387	1.846	3.633	2.641	1.912	1.874	2.122	1.067	0.057	1.959
WWW'12	1.635	1.924	0.995	2.289	1.953	1.599	1.867	1.945	1.054	2.461	0.065

Figure 4: Structural similarity values for the analyzed datasets (the upper table contains the property-based conversion (PB) values; while the lower one contains the property-class-based (PCB) values. The values of in green (the diagonal) are the compression ratios that KRIMP achieves for that particular dataset. The other values are colored on a color scale from white to red with the lowest value in white and the highest in red.

data structures have changed, and this change has appeared in the code tables. Note that this means that we could integrate the new data into the older ones and expect that globally the structure of the data is not going to change, while not the opposite. This kind of observation is particularly important when dealing with data integration. Moreover, we can also say that a query that is able to retrieve data when posed against the old datasets is likely to retrieve data when posed to the newer ones, rather than the opposite.

Analyzing further the data, we now turn our attention to the sub-matrix formed by ISWC conferences. In both conversion, we can see how ISWC'12 and ISWC'13 are really similar in both directions with structural similarity values close to 1. We observe that these two very similar datasets also have similar similarity measures when both of them are compared to another dataset. Besides, we can see how these parallelisms in the values also happen among conferences. For example, focusing on PCB (lower table) and comparing WWW'12 against ISWC'12 gives a value of 1.045. Hence, when comparing other datasets which are close to WWW'12 against ISWC'12, their values should be similar, which can be observed for example when comparing ESWC'12 (0.995 against WWW'12) to ISWC'12. Note that there will be a small variability as KRIMP applies a heuristic, and it does not achieve the optimal result.

#### 4.4 Inclusion Tests

Our final experiments aimed at observing the efficiency of our measure in the detection of inclusion of a dataset into another, using the *Inclusion Collection*.

Due to space restrictions, in Figure 5, we only show the results for ISWC'12 derived datasets and PCB conversion<sup>2</sup>. However, our observations for this table can be generalized to the rest of our experiments. The leftmost table contains the measures using ISWC'12 as reference dataset: the first column of values shows the comparison with the other datasets alone, as in Figure 4; the last column of values contains the comparison with the union of each dataset with ISWC'12. The rightmost table contains the measures using each other dataset to measure the structural similarity with ISWC'12, in the same way with the first column for the datasets alone and the last column for the unions.

Analyzing these results, when the comparison of the reference dataset and another dataset clearly indicated structural differences, the union of this dataset with the reference is still detected as different. However, as expected, a decrease of the difference measure is observed. After the union, the code table of the reference dataset is not able to codify the structures specific to the other dataset. When we compare the reference dataset to the union dataset (rightmost

<sup>2</sup>The complete results for the whole *Inclusion Collection* can be found at <http://sid.cps.unizar.es/projects/simStruct/>

Property Class Based (PCB) [ISWC'12 <= *]				Property Class Based (PCB) [* <= ISWC'12]			
EKAW'16	2.236	EKAW'16+Ref	1.405	EKAW'16	8.918	EKAW'16+Ref	1.038
ESWC'17	2.818	ESWC'17+Ref	1.972	ESWC'17	7.645	ESWC'17+Ref	1.038
ESWC'12	0.986	ESWC'12+Ref	0.921	ESWC'12	4.195	ESWC'12+Ref	1.007
ESWC'14	1.295	ESWC'14+Ref	1.136	ESWC'14	4.183	ESWC'14+Ref	1.038
ESWC'15	1.606	ESWC'15+Ref	1.260	ESWC'15	4.237	ESWC'15+Ref	1.038
ISWC'10	0.977	ISWC'10+Ref	0.914	ISWC'10	4.077	ISWC'10+Ref	1.002
ISWC'13	1.110	ISWC'13+Ref	1.061	ISWC'13	1.038	ISWC'13+Ref	1.038
LREC'08	1.025	LREC'08+Ref	0.966	LREC'08	3.388	LREC'08+Ref	1.002
WWW'11	0.998	WWW'11+Ref	0.931	WWW'11	1.874	WWW'11+Ref	1.000
WWW'12	1.045	WWW'12+Ref	0.971	WWW'12	1.867	WWW'12+Ref	1.005

[ISWC'12 <= \*] Difference measures when comparing each dataset against ISWC'12

[\* <= ISWC'12] Difference measures when comparing ISWC'12 against each dataset

+Ref: Dataset integrated with ISWC'12 (i.e., the union of both)

**Figure 5: Results of the inclusion experiments for the ISWC'12 dataset using PCB conversion.**

table), we observe a strong similarity with measures very close to 1. This means that the patterns present in the union dataset code table are able to codify almost optimally the contained dataset (i.e., it is structurally contained within the other dataset). Inclusion can also be observed in the comparison of the original data (see Figure 4) with the conference ESWC'12 being structurally included in WWW'12.

As expected from the nature of our approach, the observed asymmetry is a strong measure of structural inclusion of one dataset into another, which indeed can be exploited in many scenarios as before mentioned (i.e., updating, integrating, querying datasets). This behavior is independent of the relative size of the graphs as our approach exploits heavily the detected structure of the graphs; in our experiments the ratio of size of the inclusion datasets with their originals varies from 0.29 to 3.5. The only restriction is to have enough data so as to leverage them and obtain structural information; in the case of testing particular instances, the inclusion test should be done exploiting the KRIMP classification properties [18], but we consider this particular use case part of the future work of this paper.

## 5 RELATED WORK

The gap between data mining and the Semantic Web has been bridged in several ways, either by using data mining as a source of data for Semantic Web approaches or by using RDF data as background knowledge to enhance data mining approaches. From the beginning of the Semantic Web, the Semantic Web mining domain, a subdomain of data mining, had as goal the extraction of RDF data from external sources, as presented in the state-of-the-art by Quboa and Saraee [13]. Outside of this field, other approaches have focused on using Semantic Web technologies for knowledge discovery in databases, as presented in the survey [15]. These approaches use RDF data and ontologies to guide their data mining approaches. On the other hand, several approaches tried to adapt pattern mining approaches to the Semantic Web, such as association rule mining [9], or formal concept analysis [1]. Our approach uses data mining techniques over RDF data to assess RDF data, it is an application of pattern extraction and

pattern comparison tailored for the Semantic Web. To our knowledge, no other approach has proposed to use data mining techniques to measure structural similarity between RDF graphs.

In the Semantic Web, the notion of similarity generally concerns the notion of semantic similarity between concepts or resources, and is usually to improve knowledge mining [9], recommender systems [10], or ontology matching systems [16]. Approaches for semantic similarity, such as the ones proposed by Piao and Breslin [10], or by Meymandpour et al. [8], focus on providing semantic similarity measures between individual concepts. On the contrary, our approach focuses on global structural aspects of the datasets, providing a measure of how similar the whole datasets are. Besides, our approach makes it possible to calculate measures without having to index all the resources at every comparison: it suffices to share the code table and the resource-item index used in the conversion in order to be able to compare efficiently other datasets to ours.

Our approach is based on the same principles as the similarity measure proposed by Vreeken et al. [17]. Instead of looking for a symmetric measure (i.e., a distance), we have revisited and reinterpreted its natural asymmetry, and exploited it to be able to measure structural similarity and inclusion. Apart from adapting the algorithm and their measure to RDF graphs, we have shown the capacities of this approach regarding flexible data models. In another domain, the notion of structural similarity appears in approaches for XML document clustering [12]. These approaches use different ways to evaluate distances between XML documents, and use these distances in clustering algorithms. In particular, Piernik et al. [11] propose to count the number of shared pattern between documents as a distance. Using only the number of shared patterns disregards the actual data distribution, which is captured in our approach thanks to using the codification length of the converted databases. In fact, our approach could be adapted to their problem.

Finally, regarding our proposed conversions, we can find other approaches proposing similar abstractions of RDF graphs for different purposes. Glim et al. [3] propose an abstraction of ABoxes to characterize individuals close to our PB and PCB conversions. This abstraction describes the asserted types and relations of individual



to create a summary used for fast inferences over low-level data. Nebot and Berlanga [9] propose a conversion from RDF graph to transactions for association rules mining, based on the property paths starting from an individual. Their conversion could be adapted to our approach to evaluate the similarity according the extended neighborhood of the instances.

## 6 CONCLUSIONS AND FUTURE WORK

Structural similarity is an important property when dealing with integrating, querying, or updating knowledge bases. However, its measure requires often costly graph comparison, which makes it difficult to scale. To deal with this issue, we have proposed the *structural similarity measure*, which makes it possible to evaluate the data structures resemblance between RDF graphs exploiting well-established data mining principles.

Our approach leverages a well-known data mining algorithm to extract the most descriptive structural patterns, and uses compression to give a measure expressing the pattern sharing degree of the compared RDF graphs. It is aimed at processing graphs large enough to be used with data mining techniques, where other approaches do not scale well. In addition, by sharing the code tables, it can reuse great part of the computation required for the comparison. To transform the RDF graph to the transactional data required, we have proposed two different conversions describing the set of properties and classes used in the neighborhood of instances. Their level of detail have shown to be enough to capture differences in vocabulary usages. Our experiments have shown that our measure captures the differences between artificially modified datasets and between different versions of real datasets. Besides, thanks to its inherent asymmetry, the experiments showed its capabilities to detect inclusion of one dataset into another.

In future works, we plan to apply our measure to different problems such as data integration [6] or document clustering as in [12]. Finally, we will work on formalizing the codification of the code tables to be included as metadata describing the different datasets.

## ACKNOWLEDGMENTS

This work is partially supported by ANR project IDFRAud (ANR-14-CE28-0012), the CICYT projects (TIN2013-46238-C4-4-R and TIN2016-78011-C4-3-R), and DGA-FSE.

## REFERENCES

- [1] S. Ferré and P. Cellier. Graph-FCA in practice. In *Proc. of Intl. Conf. on Conceptual Structures (ICCS'16)*, pages 107–121. Springer, 2016.
- [2] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113–129, 2010.
- [3] B. Glimm, Y. Kazakov, T. Liebig, T.-K. Tran, and V. Vialard. Abstraction refinement for ontology materialization. In *Proc. of Intl. Semantic Web Conf. (ISWC'14)*, pages 180–195. Springer, 2014.
- [4] G. Grahne and J. Zhu. Fast algorithms for frequent itemset mining using FP-trees. 17(10):1347–1362, 2005.
- [5] H. Knublauch and D. Kontokostas. Shapes constraint language (SHACL). W3C recommendation, W3C, 2017.
- [6] P. Maillot, T. Raimbault, D. Genest, and S. Loiseau. Consistency evaluation of RDF data: How data and updates are relevant. In *Proc. of Intl. Conf. on Signal-Image Technology and Internet-Based Systems (SITIS'14)*, pages 187–193. IEEE, 2014.
- [7] R. Meusel, P. Petrovski, and C. Bizer. The WebDataCommons microdata, RDFa and microformat dataset series. In *Proc. of Intl. Semantic Web Conf. (ISWC'14)*, pages 277–292. Springer, 2014.
- [8] R. Meymandpour and J. G. Davis. A semantic similarity measure for Linked Data: An information content-based approach. *Knowledge-Based Systems*, 109:276–293, 2016.
- [9] V. Nebot and R. Berlanga. Finding association rules in Semantic web data. *Knowledge-Based Systems*, 25(1):51–62, 2012.
- [10] G. Piao and J. G. Breslin. Measuring semantic distance for Linked Open Data-enabled recommender systems. In *Proc. of Annual ACM Symp. on Applied Computing (SAC'2016)*, pages 315–320. ACM, 2016.
- [11] M. Piernik, D. Brzezinski, and T. Morzy. Clustering XML documents by patterns. *Knowledge and Information Systems*, 46(1):185–212, 2016.
- [12] M. Piernik, D. Brzezinski, T. Morzy, and A. Lesniewska. XML clustering: A review of structural approaches. *The Knowledge Engineering Review*, 30(3):297–323, 2015.
- [13] Q. K. Quboa and M. Saracee. A state-of-the-art survey on Semantic Web mining. *Intelligent Information Management*, 5(1):10, 2013.
- [14] J. Rissanen. Minimum Description Length principle. In *Encyclopedia of Statistical Sciences*. John Wiley & Sons, Inc., 2004.
- [15] P. Ristoski and H. Paulheim. Semantic Web in data mining and knowledge discovery: A comprehensive survey. *Web Semantics: Science, Services and Agents on the World Wide Web*, 36:1–22, 2016.
- [16] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Transact. on Knowledge and Data Engineering*, 25(1):158–176, 2013.
- [17] J. Vreeken, M. van Leeuwen, and A. Siebes. Characterising the difference. In *Proc. of ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'07)*, pages 765–774. ACM, 2007.
- [18] J. Vreeken, M. Van Leeuwen, and A. Siebes. KRIMP: Mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, 2011.
- [19] Z. Zhang, A. G. Nuzzolese, and A. L. Gentile. Entity deduplication on ScholarlyData. In *Proc. of Extended Semantic Web Conf. (ESWC'17)*, pages 85–100. Springer, 2017.