Instance Based Clustering of Semantic Web Resources

Gunnar AAstrand Grimnes[1], Peter Edwards[2], and Alun Preece[3]

[1] Knowledge Management Department, DFKI GmbH
Kaiserslautern, Germany
[2] Computing Science Department
University of Aberdeen, UK
[3] Computer Science Department
Cardiff University, UK
gunnar.grimnes@dfki.de, pedwards@csd.abdn.ac.uk,
a.d.preece@cs.cf.ac.uk

Abstract. The original Semantic Web vision was explicit in the need for intelligent autonomous agents that would represent users and help them navigate the Semantic Web. We argue that an essential feature for such agents is the capability to analyse data and learn. In this paper we outline the challenges and issues surrounding the application of clustering algorithms to Semantic Web data. We present several ways to extract instances from a large RDF graph and computing the distance between these. We evaluate our approaches on three different data-sets, one representing a typical relational database to RDF conversion, one based on data from a ontologically rich Semantic Web enabled application, and one consisting of a crawl of FOAF documents; applying both supervised and unsupervised evaluation metrics. Our evaluation did not support choosing a single combination of instance extraction method and similarity metric as superior in all cases, and as expected the behaviour depends greatly on the data being clustered. Instead, we attempt to identify characteristics of data that make particular methods more suitable.

1 Introduction

Currently on the Semantic Web there is an imbalance in the number of data-produces compared to the number of consumers of RDF data. The original Semantic Web vision [1] envisaged intelligent autonomous agents that could navigate the Semantic Web and perform information gathering tasks for their users, but such agents have not yet been realised. We argue that autonomous Semantic Web agents need to be capable of more sophisticated data-processing techniques than the deductive reasoning offered by existing Semantic Web inference engines, and to be able to solve personal tasks for a user on the heterogeneous and noisy Semantic Web an agent must be able to learn. We have in previous work been exploring how traditional machine learning techniques may be applied to Semantic Web data [2,3] and in this paper we focus with the particularities arising from attempting to cluster Semantic Web data. Clustering is the process of classifying similar object into groups, or more precisely, the partitioning of a data-set into subsets called clusters. The process is applicable for a large range of problems, and would make a valuable tool for an autonomous Semantic Web agent.

S. Bechhofer et al.(Eds.): ESWC 2008, LNCS 5021, pp. 303–317, 2008.

In this paper we identify two challenges that need to be tackled for applying traditional clustering mechanisms to Semantic Web resources: firstly, traditional clustering methods are based on *instances* not on a large interconnected graph as described by RDF; How do we extract a representation of *an instance* from such a RDF graph? Secondly, having extracted the instances, how do you computer the distance between two such instances?

We discuss three approaches to instance extraction in Section 2.1 and three distance measures in Section 2.2. We test our methods by using a simple Hierarchical Agglomerative Clustering (HAC) [4] algorithm. Although many more performant and modern clustering algorithms exist, we are not primarily interested in absolute performance, but rather the relative performance between our different approaches. HAC is stable, generates clustering solutions with exclusive cluster membership and generally well understood and is thus well suited for such an empirical comparison. The result of the HAC algorithm is a tree of clusters, with the similarity between clusters increasing with the depth of the tree. The tree can easily be cut to generate any number of separate clusters by iteratively cutting the weakest link of the tree until the desired number of clusters is reached.

We test our clustering techniques on three different RDF data-sets: the first consisting of computer science papers from the Citeseer citation database[1], where we generated RDF descriptions of each using the BibTeX description. The papers are categorised into 17 different subject areas of computing science, and exemplifies a typical relational database to RDF conversion: the data is shallow, there is no real ontology and most properties take literal values, and very few properties relating two resources. This dataset has 4220 instances. The second dataset is a Personal Information Model (PIMO) from a user of the Semantic Desktop system Gnowsis [5] – this data-set is based on a well-defined and rich ontology and exemplifies a data-set built on Semantic Web best practises. Both the ontology and the instances are partially generated by an end-user, but using a sophisticated tool assuring that the data is consistent and semantically sound. This dataset has 1809 instances in 48 classes. The third data-set is a crawl of FOAF documents (Friend of a Friend[2]) consisting of 3755 person instances, this data-set is identical to the one used in [2]. For the citeseer and PIMO datasets existing classifications of the instances are available, allowing a supervised evaluation. We present the results of several common metrics for evaluating clustering solution in Section 3, and Section 4 makes concluding remarks and outlines the future plans for this work.

2 Clustering RDF Instances

When applying traditional clustering techniques to RDF data there are two issues that require consideration:

1. Instance extraction – What exactly constitutes an individual in an RDF graph? The main virtue of RDF is that all data be interconnected, and all relations can be represented and made explicit. However, this means that a rich data-set might just appear

[1] Citeseer: http://citeseer.ist.psu.edu/, Our dataset can be downloaded from http://www.csd.abdn.ac.uk/∼ggrimnes/swdataset.php

[2] The FOAF Project, http://www.foaf-project.org/

as one large graph, rather than separate instances. By instance extraction we mean the process of extracting the subgraph that is relevant to a single resource from the full RDF graph. The possibilities range from considering only the immediate attributes, to more sophisticated analysis of the whole graph, based on the graph topology, ontological information or information-theory metrics.

2. Distance Measure – How is the distance between two RDF instances computed? Assuming that the output from the instance extraction is a smaller RDF graph with the resource as root node, how may one compare two such graphs? They might partially overlap, or they may only have edges with common labels. Again the alternatives range from naive approaches to methods based on graph-theory or the ontological background information.

In addition to these two points, a Semantic Web capable of autonomous clustering requires a method for determining the appropriate number of clusters for a data-set. However, this problem is not unique to Semantic Web clustering nor is the process changed significantly by the processing of RDF, and we will not discuss it further in this paper.

2.1 Instance Extraction

The problem of instance extraction is best illustrated with an example. Consider the example RDF graph shown on the left in Figure 1. We would like to extract the parts of the graph relevant to the resource *ex:bob*. Bob is related to several other RDF resources in this graph; he knows some other people, he works for a company and he is the author of publications. Several of these other resources might constitute important parts of what defines Bob as a person. The challenge is to find a reasonable subset of the graph surround Bob, that still includes any important information without being too large. The problem of extracting instance subgraphs is not exclusive to clustering. For instance, the SPARQL specification[3] includes the *DESCRIBE* directive which should return *a single result RDF graph containing RDF data about resources*. However, the exact form of this description is not specified in the current SPARQL specification, and should instead be determined by the SPARQL query processor. Another area where these methods are applicable is programming user-interfaces for RDF enabled applications, where it is often required to extract an informative representation of a resource for presentation to an end-user. We will discuss three domain-independent approaches that have been implemented for this paper, and we will revisit the example graph for each one. In addition we will discuss some possible domain-dependent methods that improve on the domain-independent methods, but require additional semantic infra-structure support that is currently unavailable on the Semantic Web.

Immediate Properties. Our first and most direct approach for extracting instance representations from RDF graphs is considering only the immediate properties of resources. This method is illustrated on the right of Figure 1 where the extracted part of the graph around the *ex:bob* resource has been highlighted. This seems like an intuitive approach if viewing RDF from an object-oriented programming perspective, it is simple to implement and it runs very quickly. However, the approach has the obvious drawback that

[3] The SPARQL Specification: http://www.w3.org/TR/rdf-sparql-query/

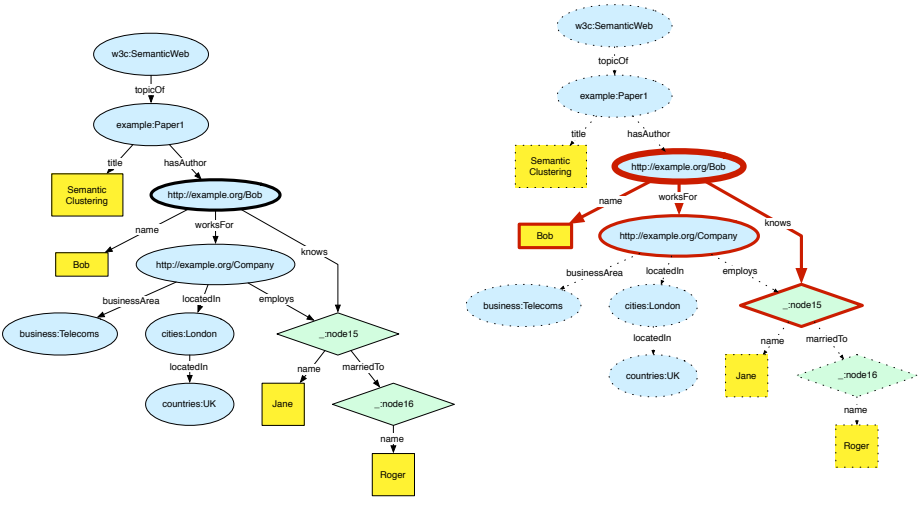

Fig. 1. An Example RDF Graph and Naive Instance Extraction

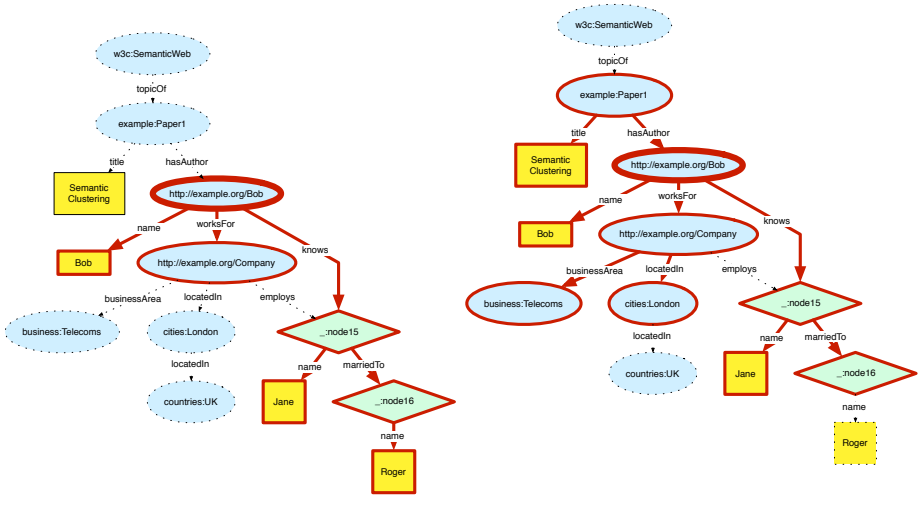

Fig. 2. CBD and DLT Instance Extraction Examples

much of the information that is relevant to the resource is lost, indeed large parts of the graph may not be included in any instance graph.

Concise Bounded Description. An improvement over simple immediate properties can be achieved by taking the types of nodes in the graph into account. Concise Bounded Description (CBD)[4] is one such improvement and is defined as a recursive algorithm

[4] Concise Bounded Description: http://www.w3.org/Submission/CBD/

for extracting a subgraph around an RDF resource. Informally it is defined as the resource itself, all it's properties, and all the properties of any blank nodes connected to it, formally the CBD of graph G is defined recursively as follows:

$$\begin{aligned} CBD_G(x) = &\{< x, p, o > \mid < x, p, o > \in G\} \\ &\cup \{< s, p, o > \mid < s, p, o > \in G \\ &\wedge \exists\, s', p' (< s', p', s > \in CBD_G(X) \wedge bn(s))\} \end{aligned}$$

where $bn(x)$ is true if x is a blank node.

The CBD extraction from the example graph is shown on the left in Figure 2, where the extracted graph is highlighted. Note how all info about *_:node15* and *_:node16* is included, but no further triples from *http://example.org/Company*. An alternative CBD definition also considers backward links, where triples having the resource in question as an object are also included. CBD is a much better option than just considering the immediate properties of a resource, however it also has several disadvantages: since the behaviour of CBD depends on the use of blank nodes in the data it is not strictly domain independent. For instance, some applications discourage the use of blank nodes because of the difficulty in implementing their semantics correctly, in this case CBD is no better than immediate properties. Conversely, in data where no natural identifiers exist there might be a large ratio of blank nodes to labelled resources, and the CBD might only be bound by the size of the full graph.

Depth Limited Crawling. An alternative to limiting the sub-graph by node-type is to limit the sub-graph by depth and simply traverse the graph a certain number of steps from the starting node. This has the advantage that it is stable over any input data making it easier to limit the size of the subgraph. For our initial experiments with clustering of FOAF data [2] we found that traversing two edges forward and one edge backwards from the root node provides a good tradeoff between size and the information contained in the subgraph, although this is clearly data-dependent and may need adjusting for special cases. The sub-graph extracted from the example graph is shown on the right in Figure 2. Note how this is the only method where also the publications that link to *Bob* as the author are included.

Since the results of the instance extraction algorithms vary greatly depending on the input data, we analyse the graphs extracted from our three data-sets. The mean and standard deviation of the number of triples in the graphs extracted is shown in Table 1. Note that the results for the Citeseer dataset are the same for all extraction methods, the reason being that the only properties linking two resources in this dataset is *rdf:type* relating publications to their class, and the classes have no further properties. All other properties in this dataset are literal properties. Note also that the naïve approach and CBD is identical for the PIMO data-set, as the PIMO language disallows the use of blank nodes, so CBD does not do any further traversal. The table also clearly illustrates the weakness of CBD when dealing with large crawled data, since FOAF encourages the use of blank nodes to represent people it is possible to traverse large parts of the FOAF crawl through blank nodes alone. In fact, the largest sub-graph extracted from the FOAF graph using CBD contained 20108 triples.

Table 1. Mean and standard deviation of size of extracted instance graphs

Dataset	Naive	CBD	DLT
Citeseer	18.22 / 5.31	18.22 / 5.31	18.22 / 5.31
PIMO	6.84 / 1.49	6.84 / 1.49	828.04 / 5.12
FOAF	13.95 / 19.14	1299.26 / 2983.97	145.06 / 153.58

2.2 Distance Measure

The choice of distance measure is the most important factor for a clustering setup, the definition of what constitutes close or remote instances affects everything about the behaviour of the clusterer. Commonly clustering has been performed on instance represented as numerical vectors and viewing the vectors as coordinates in an N-dimensional space allows the use of geometric distance measures, such as Euclidean or Manhattan distance.

However, transforming RDF instances represented as graphs into coordinates in a Euclidean space is not trivial, and thus standard distance metrics do not readily apply to RDF instances. In this section we present several methods for solving the problem of finding the distance between pairs of RDF instances, each instance consisting of a root node (the RDF resource) and a (sub)-graph considered relevant to this instance, known as R_n and G_n respectively. We consider one distance measure inspired by traditional feature vector based clustering, one based on the topology of the RDF graphs and one based on the ontological background information available.

Feature Vector Based Distance Measures. As a base-line approach to an RDF similarity measure we wanted to adapt traditional vector-based similarity to comparing RDF instances. The first step is to transform an RDF graph and root node pair into a feature vector representation. Since the method should be compatible with all instance extraction methods outlined above we needed to go beyond the obvious solution of mapping the direct RDF properties of the resource to features in the vector. Instead we map paths in the RDF graph to features, and let the value of the feature be a set of all the nodes reachable through that path. First we defined the set of nodes reachable from a root-node X in a graph G:

$$reachable(X, G) = \{n| < x, p, n >\in G\} \cup \{n'|n' \in R_G(n)\} \quad (1)$$

The feature vector for a single instance n is then defined as:

$$FV(n) = \{shortestPath(R_n, x, G_n)|x \in reachable(R_n, G_n)\} \quad (2)$$

where $shortestPath(R, X, G)$ is the shortest path from node R to node X in graph G, and a path is defined as a list of edges to traverse (i.e. RDF properties). The feature vector for a set of instances is simply the union of all the individual feature sets. Consider again the example graph in Figure 1, the features for the instance *ex:bob* would be as follows:

Table 2. Longest and average path-lengths

Dataset	Naive	CBD	DLT
Citeseer	1.00 / 0.98	2.00 / 1.02	2.0 / 1.20
PIMO	1.00 / 0.97	1.00 / 0.97	3.0 / 1.87
FOAF	1.00 / 1.00	26.00 / 19.40	3.0 / 1.90

```
[ name, worksFor, knows, worksFor→businessArea, worksFor→locatedIn,
knows→name, knows→marriedTo, worksFor→locatedIn→locatedIn,
knows→marriedTo→name ]
```

Note how the path `worksFor→employs` is not included as the same resource (*_:node15*) can be reacher by the shorter path of `knows`. As the feature vector can get very long for complicated data-set it is often desirable to only include the most frequent paths. In our experiments we used only the 1000 paths most frequently included when considering all instances, i.e. we used the same feature vector of length 1000 for all instances, but not all features have values for all instances. Table 2 shows the maximum and average length for the feature vectors extracted from our three data-sets when using each of the three instance extraction methods above; once again the weakness of CBD combined with FOAF is clear, but no other extraction method or dataset result in prohibitively long vectors. As mentioned above, the values of each feature is the set of RDF nodes reachable through that path[5]. It is necessary to use set-valued features because each RDF property might be repeated several times. One could imagine that cardinality constraints from an ontology could be used to restrict certain paths to single valued features, however, such constraints are unlikely to remain unbroken on a noisy heterogeneous Semantic Web, and we are primarily interested in methods that do not depend on data correctly conforming to an ontology. Again, consider the example graph from Figure 1, the values of the feature vector for the instance *ex:bob* would be:

```
[ {``bob''}, {ex:TheCompany}, {_:node15}, {business:Telecoms}, {cities:London},
{``Jane''}, {_:node16}, {countries:UK}, {``Roger''} ]
```

Note how in this basic examples every feature is a single valued set and no features are missing. In a more complex scenario many of the feature-values might be the empty set where the RDF properties in the path are not available. Computing the distance between two feature vectors is then done using a straight-forward similarity measure inspired by the Dice coefficient: The distance between instance X and Y over feature-vector FV is defined as:

$$simFV(X, Y, FV) = \frac{1}{|FV|} \sum_{f \in FV} \frac{2 * |X_f \cap Y_f|}{|X_f| + |Y_f|} \tag{3}$$

[5] Note that this is not quite as query intensive as it may seem, by collapsing the common subpaths in all paths to a tree, an efficient algorithm can be devised that makes only the minimal set of queries.

where X_f and Y_f is the set value of feature f for X and Y respectively. The $simFV$ takes a value between 0 and 1 where 1 means that the instances have exactly the same properties and 0 means no shared properties.

Graph Based Distance Measures. Montes-y-Gómez et. al developed a similarity measures for comparing two conceptual graphs [6]. Conceptual graphs are a data-structure commonly used for natural language processing. They consist of a network of concept nodes, representing entities, attributes, or events and relation nodes between them. The similarity metric incorporates a combination of two complementary sources of similarity: the *conceptual similarity* and the *relational similarity*, i.e. the overlap of nodes and the overlap of edges within the two graphs.

Conceptual graphs and RDF instances are structurally sufficiently similar that the this similarity metric is also appropriate for comparing RDF graphs. For each instance to be clustered the extracted sub-graph is used for the comparison and the root node is ignored. Space reasons prevent us from presenting the details of the metric here, please refer to Montes-y-Gómez et. al's paper for the details and to our previous work for details on applying this metric to RDF data [2].

Ontologically Based Distance Measures. There have been several efforts for developing an ontology based similarity metric for RDF data. Most of these originate from the fields of ontology mapping and are focussing on similarity between whole ontologies, or classes within. For instance, [7] compares two ontologies using an approach based on conceptual graphs, and [8] presents a metric designed especially for OWL-Lite. Maedche and Zacharias have developed a similarity metric explicitly for instance data, but with heavy emphasis on the ontological background information [9] . Their metric assumes the existence of a well-defined ontology and conforming instance data. However, real Semantic Web data is often noisy and inconsistent and some minor adjustments to the algorithm are required to allow for handling of more realistic data. The ontological similarity metric is a weighted combination of 3 dimensions of similarity:

$$simOnt(X,Y) = \frac{t \times TS(X,Y) + r \times RS(X,Y) + a \times AS(X,Y)}{t + r + a} \tag{4}$$

Where TS is the taxonomy similarity, RS the relational similarity, AS the attribute similarity, and t, r, a the weights associated with each. The taxonomy similarity considers only the class of the instances being compared and is determined by the amount of overlap in the super-classes of each instance's class. The attribute similarity focuses on the attributes of the instances being compared, i.e. the properties that take literal values (known as data-type properties in OWL), and is based on using some type-specific distance measure for comparing the literal values for matching predicates (i.e. Levenshtein edit distance for string literals and simple numeric difference for numbers, etc.) Finally, the relational similarity is based on the relational properties of a resource, i.e. the properties that link this resource to other resources (and not to literals). This is defined recursively, so for finding the similarity between two linked resources the ontological similarity measure is applied again. To make it possible to compute the similarity measure the recursion is only followed until a certain depth, after which the relational similarity is ignored and the combination of only attribute similarity and taxonomy similarity is returned. For our experiments we stopped recursing after 3 levels. Again we

do not have the space to present the algorithm in full, and we will only briefly cover our modifications that allowed us to relax some assumptions made about the input data which are unlikely to hold true for real-life Semantic Web data, namely:

1. The ontologies must be strictly hierarchical, i.e. no multi-class inheritance.
2. Resources must be members of exactly one class.
3. Range and domain must be well defined for all properties.
4. Range and domain constraints must never violated.
5. There must be a clear distinction between datatyped properties and object properties, i.e. a property takes either literal objects or other resources.

Both point one and two are natural simplifying restrictions many applications chose to enforce internally, although both multi-class inheritance and multiple types are supported by both RDFS and OWL. Point three is not required by RDF, but would be a logical effect of using well-engineered ontologies. Point four is a valid assumption if one assumes a semi-closed data-base view of the Semantic Web (i.e. the database-semantic web view). Point five is another natural assumption to make since in OWL the distinction is made between datatype properties and object properties (Maedche and Zacharias call these attributes and relations). However, for RDFS based ontologies this is not the case.

To enable $simOnt$ to work with our data-sets we made the following modifications to the algorithm:

1. For the definition of the taxonomy similarity we let $SC(x)$ mean the set of super-classes of ALL types of x.
2. We do not rely on range/domain alone for finding applicable literal-/relational properties, we also add the properties that are used with literals/resource in the data available. This also solves the missing range/domain declaration problem.

Note that this distance measure does not make use of the extracted sub-graph, only the ontological information associated with the root-node.

3 Comparison

Clustering is a task that is notoriously hard to evaluate. Recent overviews over available methods can be found in [10]. Ultimately clustering is a data-analysis exercise, and the true value of a clustering solution comes from understanding of the data gained by a human data-analysist, but in this paper we are interested in automated methods for checking the quality of a clustering solution. For two of our data-sets the preexisting classifications may be used for evaluation, but since there is no way to train a clustering algorithms to look for particular classification of the data, a supervised evaluation with regard to this classification is not necessarily a good measure of the performance of the clusterer. I.e. the clustering algorithm might have chosen to partition the data along an axis different to partitioning in the given "correct" classification, meaning that although the supervised quality measures might be very poor, the resulting clustering might be an equally correct classification. In this paper we conduct a supervised evaluation for the PIMO and citeseer using the following supervised quality metrics:

- Entropy — A supervised measure that looks at how the different classes are distributed within each cluster. If each cluster only contains instances from a single class the entropy is zero. The entropy is trivially optimised by having a single cluster containing all instances.
- Purity — A supervised measure that quantifies the degree of which a cluster contains instances from only one class. As for entropy, this is trivially optimised by having a single cluster containing all instances.
- F-Measure — The F-measure in information retrieval is an harmonic mean of precision and recall, and can also be used a supervised quality metric for clustering [11]. For a perfect clustering solution with an F-Measure of one there must be a cluster for each class, containing exactly the members of the corresponding class. This is unlikely for an unguided clustering solution, and one can therefore not compare F-measure values for clustering with what is considered a good performance for classification algorithms.
- Hess Precision/Recall — Another view of precision and recall for clustering was introduced in a recent PhD thesis by Andreas Hess [12] . Instead of considering individual instances when defining precision and recall, Hess considers all pairs of objects that were clustered. The pairs are then classified whether they occur in the same class in the correct classification as in the clustering solution, and precision and recall is then defined as normal for each pair. Hess observes that the precision metric is biased towards small clusters and the recall towards large clusters.

For the FOAF and PIMO data-set we also carried out an unsupervised evaluation, using the basic Sum of Squared Errors and Zamir's Quality Metric (ZQM) [13], a function aiming to represent a trade-off between small well-defined clusters and larger mixed clusters.

3.1 Results

We evaluated all combinations of the instance extraction methods and similarity metric presented above on the PIMO and Citeseer data-set. For both data-sets we specified the number of clusters to generate based on the preexisting classification and calculated the entropy, purity, F-measure and Hess's pair-wise precision and recall based on this classification. The results are shown in Table 3 and the F-Measure and Hess-Recall are shown graphically in Figure 3 (for space reasons plots were only included for the most interesting metrics).

For the FOAF and PIMO data-set we used the jump-method as described by Sugar and James [14] to find a natural number of clusters for each data-set and user the unsupervised evaluation metrics to analyse the resulting clusters. The number of clusters found and evaluation metrics are shown in Table 4, the size of clusters found and the ZQM are also shown graphically in Figure 4.

Note first that recreating the preexisting classification is a very challenging problem, especially for the Citeseer dataset where the RDF representation of the papers is very simple and the information is not really sufficient for recreating the subject-area classification. Note also that it might also be possible to achieve overall better results using

[6] Please note the non-zero origins of F-Measure graphs and different scale for the two Hess Recall graphs.

Table 3. Results for Supervised Evaluation of PIMO and Citeseer Data-sets

Data-set	Extract.	Sim.	Entr.	Purity	F-Meas.	Hess-P	Hess-R
citeseer	N/A	ont	0.96	0.11	0.12	0.07	0.22
citeseer	naive	cg	0.94	0.12	0.13	0.07	0.36
citeseer	dlt	cg	0.94	0.12	0.13	0.07	0.38
citeseer	cbd	cg	0.94	0.13	0.13	0.07	0.53
citeseer	dlt	fv	0.97	0.09	0.12	0.07	0.78
citeseer	naive	fv	0.97	0.09	0.12	0.06	0.89
citeseer	cbd	fv	0.97	0.09	0.12	0.07	0.89
pimo	ont	ont	0.32	0.59	0.42	0.38	0.28
pimo	naive	cg	0.32	0.60	0.42	0.38	0.28
pimo	dlt	cg	0.31	0.60	0.44	0.40	0.29
pimo	cbd	cg	0.31	0.59	0.43	0.39	0.29
pimo	dlt	fv	0.30	0.62	0.44	0.40	0.31
pimo	naive	fv	0.33	0.59	0.42	0.39	0.33
pimo	cbd	fv	0.33	0.58	0.43	0.40	0.37

Table 4. Results for Unsupervised Evaluation of FOAF and PIMO Data-sets

Data-set	Extract.	Sim.	SSE	ZQM	# Clusters.
pimo	naive	fv	61.74	0.06	101.
pimo	naive	cg	120.97	0.08	104.
pimo	cbd	fv	59.67	0.06	117.
pimo	cbd	cg	126.38	0.15	70.
pimo	dlt	fv	35.77	0.04	128.
pimo	dlt	cg	3.91	0.04	96.
pimo	N/A	ont	32.11	0.5	16.
foaf	naive	fv	47.09	0.06	120.
foaf	naive	cg	65.7	0.09	86.
foaf	cbd	fv	77.95	0.18	56.
foaf	cbd	cg	60.62	21.43	2.
foaf	dlt	fv	35.98	0.05	113.
foaf	dlt	cg	135.69	0.13	95.
foaf	N/A	ont	29.14	0.52	14.

a more sophisticated clustering algorithm than HAC, but as mentioned previously, this was not the focus of this work.

Looking first at the unsupervised results, observer that for the PIMO data-set the number of clusters detected are far higher than the number of classes in the existing classification (The PIMO data-set has 48 classes). This holds for all methods apart from the ontological similarity measure. This can be explained by the idiosyncratic *Person* class in the PIMO data-set, containing 387 instances. We believe that further sub-divisions of the instances of this class are possible, making a larger number of clusters more natural for the PIMO dataset. Something that is not clear from the result table or graphs is that although many clusters are generated, many of them are singleton clusters, this holds even for the supervised clustering results where the *right* number of clusters is specified. The feature-vector distance measure was especially prone to creating a small number of large clusters, for the Citeseer dataset all experiments with the feature-vector distance metric the largest clusters contained over 85% of the instances and only 2 or 3 clusters had more than a single instance. For the PIMO dataset every method generated one large cluster with roughly half of the instances, caused again by the abnormal

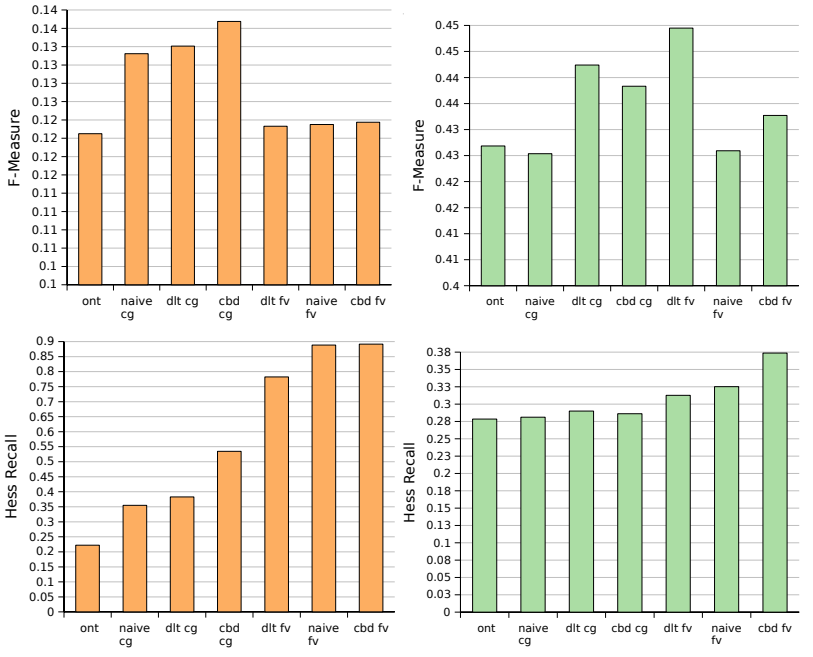

Fig. 3. F-Measure and Hess-Recall results for the Citeseer (left) and PIMO (right) data-sets[6]

Person class, the PIMO instances in this class are automatically generated from the user's address-book, and these all have identical structure.

For the FOAF data-set the number of clusters detected varies widely between the different methods. The two values that stand out are for the run using CBD and the conceptual graph similarity measure, which is extremely low, only two clusters were found. As mentioned before, the FOAF data-set exposes a problem with CBD due to the large number of blank-nodes and for some instances very large sub-graph are extracted. In combination with CG these huge instance graphs cause very unusual results. For the feature vector similarity metric the problem is not so severe, as the large instance graphs will only contribute a few features, and these are likely to be excluded from the 1500 most common features used in the end.

Considering the supervised results, we know that the PIMO dataset has many more classes than the Citeseer dataset (48 to 17), but fewer instances (1809 to 4220) and this is clearly reflected in the scores for entropy and purity, the PIMO clustering solutions has many classes that contain only a single instance which gives a high purity score, whereas for the Citeseer data-set each cluster is likely to have a mix of instances and the entropy is therefore high. Note also that for the Citeseer data-set the choice of instance extraction mechanism is largely irrelevant because the data is so shallow (there are no sub-graphs deeper than a single level).

Looking at the graphs in Figure 3 we can see that surprisingly the feature-vector method achieved high Hess-Recall scores for both data-sets, however, this is a side-effect of the large clusters created by this method and is consistent with Hess'

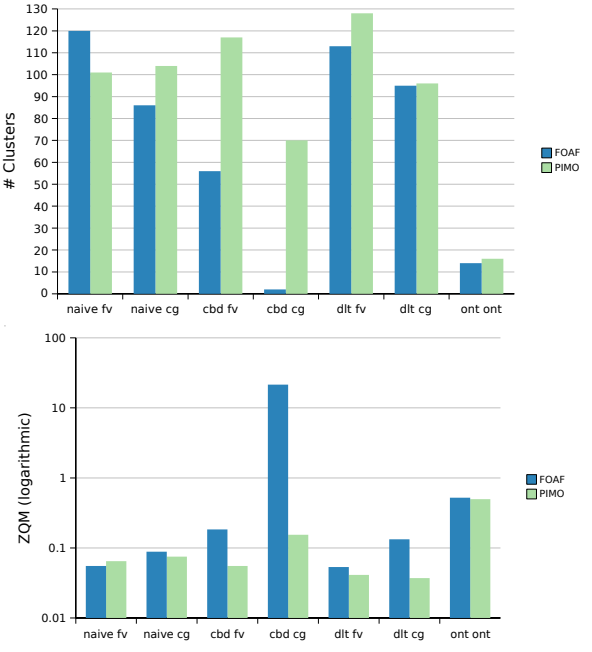

Fig. 4. Cluster Sizes and Zamir Quality Measure FOAF and PIMO data-sets

observation. Looking at the F-measure for the feature-vector it is consistently low for the Citeseer data-set, but performs rather well for the PIMO data-set, referring back to Table 2 we can see that the Citeseer data-set is just not rich enough to allow for paths of any length and any real distinction between the instances is lost.

The ontological similarity measure performs consistently well in these experiments, but not significantly better than the others. It does however seem to result in a much lower number of clusters being generated, which also results in a higher ZQM score due to the part of ZQM favouring a small number of well-defined clusters. The ontological similarity measure is theoretically well founded and makes the most use of the semantics out of the similarity measures presented here, and we had therefore expected it to perform better, but it may be that our data-set have insufficient ontological data to really make the most of this metric. However, it must be noted that the run-time performance of this metric is much worse than the two other metrics that were investigated; the recursive nature often makes it necessary to compute the similarity between a much larger number of nodes than the instances that are going to be clustered in the first place: For instance, clustering a small set of only 8 FOAF person nodes, described by only 469 triples, 178 nodes were visited. On larger graphs this number can grow very quickly. On the other hand, an advantage of the ontological similarity measure is that extraction of instance-graphs from the large RDF graph is not required, removing one layer of complexity from the clustering process.

In summary, the results do not support a clear conclusion of what is the *best* combination of extraction method and distance metric, and the best performing combination

varies for each data-set. The DLT extraction method may be the best choice for a data-independent extraction method, as although it may occasionally excluded parts of the data-graph it has no obvious weakness like CBD has for FOAF. For similarity measure the ontological similarity does well for unsupervised clustering, generating a small number of well-defined clusters, but at the cost of a longer running-time. For supervised clustering it performs worse since not enough clusters are generated to mirror the preexisting classification.

4 Conclusion and Future Work

In this paper we have identified two important challenges for performing clustering of Semantic Web data, firstly extracting instance from a large RDF graph and secondly computing the distance between these instances. We evaluated our approaches using three datasets representing typical data one might find on the Semantic Web. Our evaluation did not yield one combination of instance extraction method and distance measure that outperforms the others, but highlights a strong dependency on the structure of the data for choosing the optimal methods.

An additional method of instance extraction we would like to investage is the creation of a hybrid method for instance extraction, combining CBD and depth-limited traversal, where one always traverses the graph a certain minimum number of steps regardless of the node-type, then follow only blank-nodes up to a certain maximum level. Another approach would be to consider the frequency of the predicates when crawling, for instance an algorithm could be conceived that would only include triples not included in any other immediate sub-graph.

With regard to the evaluation, it is clearly very difficult to do a domain-independent evaluation like we attempted here. Although many metrics are available, the supervised metrics may not measure the quality we are interested in, and the unsupervised metrics are often biased towards one particular shape of clusters, i.e. either many small clusters or few large ones. A more interesting evaluation method we would like to explore is to use classification algorithms to learn a classifier for each of the identified clusters, and the clustering quality becomes the accuracy of the classifiers learned.

Finally, we believe that we have shown that clustering of RDF resources is an interesting area, where many questions remain unanswered. Our long-term goal is to create a tool-box of learning techniques that an autonomous Semantic Web Agent may use to understand the world; and we believe clustering could form a central tool for such an agent. Further investigation is needed to determine how shallow data-analysis could help an agent chose the optimal instance extraction methods, distance metrics and clustering algorithms.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284, 28–37 (2001)
2. Grimnes, G.A., Edwards, P., Preece, A.: Learning Meta-Descriptions of the FOAF Network. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 152–165. Springer, Heidelberg (2004)

3. Edwards, P., Grimnes, G.A., Preece, A.: An Empirical Investigation of Learning from the Semantic Web. In: ECML/PKDD, Semantic Web Mining Workshop, pp. 71–89 (2002)
4. Johnson, S.C.: Hierarchical clustering schemes. Psychometrika 2, 241–254 (1967)
5. Sauermann, L., Grimnes, G.A., Kiesel, M., Fluit, C., Maus, H., Heim, D., Nadeem, D., Horak, B., Dengel, A.: Semantic desktop 2.0: The gnowsis experience. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, Springer, Heidelberg (2006)
6. Montes-y-Gómez, M., Gelbukh, A., López-López, A.: Comparison of Conceptual Graphs. In: Cairó, O., Cantú, F.J. (eds.) MICAI 2000. LNCS, vol. 1793, pp. 548–556. Springer, Heidelberg (2000)
7. Dieng, R., Hug, S.: Comparison of personal ontologies represented through conceptual graphs. In: Proceedings of ECAI 1998, pp. 341–345 (1998)
8. Euzenat, J., Valtchev, P.: An integrative proximity measure for ontology alignment. In: Proceedings of the 1st Intl. Workshop on Semantic Integration. CEUR, vol. 82 (2003)
9. Maedche, A., Zacharias, V.: Clustering ontology-based metadata in the semantic web. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 348–360. Springer, Heidelberg (2002)
10. Strehl, A.: Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining. PhD thesis, The University of Texas at Austin (2002)
11. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: KDD 1999: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 16–22. ACM Press, New York (1999)
12. Heß, A.: Supervised and Unsupervised Ensemble Learning for the Semantic Web. PhD thesis, School of Computer Science and Informatics, University College Dublin, Dublin, Ireland (2006)
13. Zamir, O., Etzioni, O., Madani, O., Karp, R.M.: Fast and intuitive clustering of web documents. In: KDD, pp. 287–290 (1997)
14. Sugar, C.A., James, G.M.: Finding the Number of Clusters in a Data Set - An Information Theoretic Approach. Journal of the American Statistical Association 98, 750–763 (2003)