Social Tags and Linked Data for Ontology Development: A Case Study in the Financial Domain

Andrés García-Silva
Ontology Engineering Group,
Universidad Politécnica de
Madrid, Spain
hgarcia@fi.upm.es

Leyla Jael García-Castro
Department of Computer
Languages and Systems,
Universitat Jaume I, Castellón
de la Plana, Spain
leylajael@gmail.com

Alexander García
LinkingData I/O LLC,
Fort Collins, 80524, CO, USA
alexgarciac@gmail.com

Oscar Corcho
Ontology Engineering Group,
Universidad Politécnica de
Madrid, Spain
ocorcho@fi.upm.es

ABSTRACT

We describe a domain ontology development approach that extracts domain terms from folksonomies and enrich them with data and vocabularies from the Linked Open Data cloud. As a result, we obtain lightweight domain ontologies that combine the emergent knowledge of social tagging systems with formal knowledge from Ontologies. In order to illustrate the feasibility of our approach, we have produced an ontology in the financial domain from tags available in Delicious, using DBpedia, OpenCyc and UMBEL as additional knowledge sources.

1. INTRODUCTION

Ontology development methodologies and methods were initially based on centralized ontology development settings. In contrast, more recent proposals, e.g., DILIGENT [21], the Human-Center Methodology (HCOME) [18], and the NeOn Methodology for Building Ontology Networks [25] mainly focus on decentralized settings, emphasizing the need for community engagement in the ontology creation process as well as the constant evolution of the knowledge represented in ontologies. One of the main challenges to overcome in this decentralized context, where different teams collaborate in the ontology development, is that of knowledge acquisition. This process requires harnessing the knowledge of a group of experts and users to extract domain knowledge and represent it in the form of an ontology.

In parallel to this evolution in the area of ontology development, Social Tagging Systems (STS) have shown great success in encouraging users to create, tag, and share content, what leads to the emergence of folksonomies from user annotations. Folksonomies are lightweight conceptual structures that associate users terms (tags) to Web resources, facilitating their classification and organization [14]. Within STS the vocabulary used to tag tends to stabilize over time [11] since using existing tags is more frequent than generating new ones [24]. Facilitating in this manner the standardization in the use of a particular term in relationship to a tagged resource. It has also been reported that tag-based vocabularies from socially connected users, communities of practice, overlap more significantly than those of randomly selected users [19].

These emergent vocabularies turn folksonomies into interesting knowledge sources from which ontologies can be developed. In fact, several approaches have been proposed for deriving ontologies from folksonomies, as described in Section 2. However, results remain limited, given the inherent problems of polysemy, ambiguity, and misspelling that are present in user-based annotations. Besides, most of the approaches focus on delivering clusters or hierarchies of terms rather than ontologies; and the distinction between classes and instances is restricted.

In our approach, we share the view of eliciting knowledge from folksonomies, but go a step further, relating them to existing knowledge bases in the Linked Open Data cloud [4]. We generate a baseline domain ontology from a folksonomy in a STS, *i.e.* draft version containing an initial vocabulary organized within a structure of classes and rela-

tionships. We emphasize that it is a baseline ontology in the sense that it is possible that domain experts and ontology engineers need to prune or complete the ontology at the end of the process. We think our approach may save time to ontology engineers since the ontology is automatically generated and allows them to understand the domain at hand with a limited participation of domain experts.

Our method starts by selecting the part of the folksonomy that we will initially process: we use relevant Web sites previously identified by domain experts, then we retrieve annotations from those sites and related sites in the folksonomy following a spreading activation strategy [7]. The initial terminology is then refined; classes and relationships are identified by examining that controlled vocabulary against existing knowledge bases. This method can be seen as the process to extract domain knowledge from large and generic knowledge bases which is driven by the domain terminology in the folksonomy. Benefits of this selection includes smaller and more focused ontologies which are potentially easier to understand and to maintain. In addition, complex queries and reasoning task may execute faster on smaller data sets.

In observance of methodological practices, our technique harvests community knowledge and reuses existing ontologies. As a result of the application of our method a domain ontology is generated; with the advantage that such ontology has also links to external classes and relationships available in the Linked Open Data cloud.

To illustrate the feasibility of our approach we present a case study where we generate an ontology in the financial domain by first harvesting domain terminology from annotations in Delicious[1], and then enriching semantically that terminology with classes and relationships contained in DBpedia [5], OpenCyc[2] and UMBEL[3]. We evaluate this ontology using a gold standard ontology, and analyze the content of the ontology with the help of automatic tools such as the partitioning tool PATO [23] to identify modules in the ontology, and SWAT[4] to express the ontology in natural language sentences so that domain experts can easily evaluate the ontology content.

This paper is organized as follows. In section 2 we present the related work. Next, in section 3, we describe our approach to generate domain ontologies from folksonomies and linked data. Then, in section 4, we present the evaluation and analysis of ontologies in the financial domain generated with our method. Finally we present our conclusions.

2. RELATED WORK

Approaches aiming to obtain ontologies from folksonomies can be classified in two categories: approaches tapping into the network structure of folksonomies [3, 13, 20, 14] which rely on tag similarity measures [6, 17], and approaches using lexical resources or ontologies to specify the meaning of tags [1, 2]. Some of these approaches rely on specific techniques such as clustering [3], social networks metrics [13], formal concept analysis [14], mappings between tags and lexical resources [1] or ontologies [2], while others use a combination of them [20]. For an extensive review of the state of the art regarding folksonomies as source of knowledge we refer the reader to [10].

In [3] authors explore different clustering techniques to find related tags. They propose a graph of co-occurrent tags that is partitioned by using cluster techniques; tags are listed in decreasing order to facilitate the selection of the top N similar tags. Heymann and Garcia-Molina [13] use social networks metrics to produce a hierarchy of tags according to the similarity between tags. Similarity is calculated based on co-occurrence of tags when annotating resources. Mika [20] creates networks of tags and clusters relying on co-occurrence of tags. Tags are arranged in hierarchy of broader and narrower terms. Trias [14] is a semi-automatic approach that apply formal concept analysis to mine common conceptualizations. Conceptualizations are defined by groups of users that have used some tags to annotate a group of resources. These conceptualizations are used to create a hierarchy of tags.

On the other hand, Folk2Onto [1] propose to use WordNet to map tags to synsets so that they can filter out tags, identify synonyms, and assign categories to tags. This process is semi-automatic since the system must be supervised to learn how to map automatically. In [2] the semantics of tags is defined by associating them with ontology entities.

In short, all of these approches generate group of tags that may or may not be hierarchically organized. However, determining the nature of the relationships between tags is difficult. The semantics of tags and of relationships between them is not always explicitly defined [3, 13] albeit it is sometimes suggested [14], or defined in a limited scope [1, 2].

In contrast to the aforementioned approaches our method produces a domain ontology that consists of classes and defined relationships among them, including hierarchical and non-hierarchical. Our approach makes explicit the meaning of the relationships among the classes of the ontology by reusing existing conceptualizations in the linked data cloud.

3. GENERATING LIGHTWEIGHT ONTOLOGIES

As shown in figure 1 our approach takes as input a set of seeds in the form of appropriate domain resources, and produces as output the corresponding domain ontology. First we obtain a domain terminology by traversing a graph derived from the folksonomy. Seeds are used as starting points to traverse the folksonomy structure, and as reference resources to search for similar resources in the folksonomy. Next we identify the semantics of the domain terminology by reusing ontologies in the Linked Data cloud. To do so, we relate terms to resources in DBpedia [5], which is considered as a data hub in the linked data cloud given the high degree of interconnection with other data sets. Then we look for classes that are related to those DBpedia resources. These classes are not necessarily in the DBpedia ontology; they can be in other knowledge bases linked to DBpedia. In addition, we look for relationships among those classes in DBpedia and interconnected datasets. As output we produce a domain ontology consisting of classes and relationships that are directly linked to classes in the linked data cloud.

In fact this process can be viewed as the selection of domain knowledge from large generic knowledge bases which is driven by the domain terminology gathered from the folksonomy. Benefits of this selection include a smaller and focused knowledge base which is potentially easier to maintain and

[1]https://delicious.com/
[2]see http://sw.opencyc.org/
[3]see http://www.umbel.org/
[4]http://swat.open.ac.uk/tools/

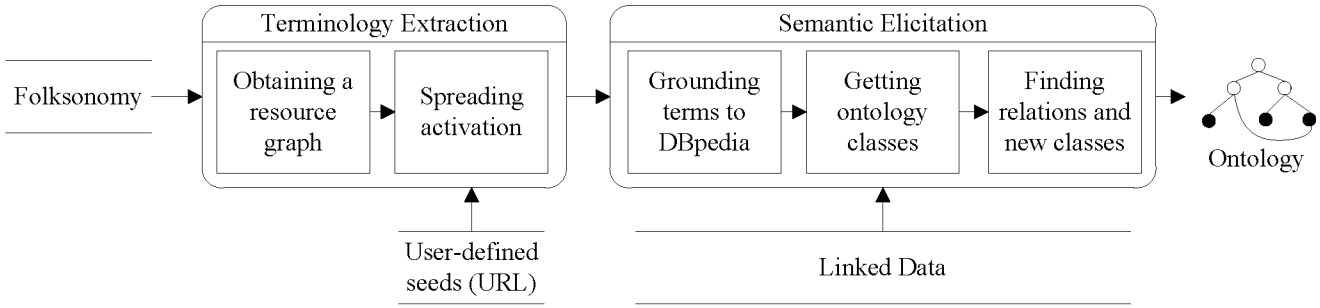


Figure 1: Eliciting knowledge from folksonomies.

tune for query performance, and that poses less cognitive burdens for users since a smaller amount of information is easier to explore and understand. In addition, in smaller knowledge bases it is possible to run reasoners in not excessive times given their current performance limitations [26]. Thus ontology engineers would be able to check the ontology consistency and coherence, or inferring new knowledge [15]. Finally since the classes are linked to classes in the linked data cloud engineers still may benefit of the interlinked information.

3.1 Terminology Extraction

A folksonomy can be understood as a set of annotations $A \subseteq U \times T \times R$ where U, T and, R are sets whose members are users, tags and resources, respectively. Thus a folksonomy can be represented as an undirected tri-partite hyper-graph $G = (V, E)$ where $V = U \cup T \cup R$, and $E = \{(u,t,r)|(u,t,r) \in A\}$ [20]. We are particularly interested in a graph $G' = (V', E')$ whose vertices V' are the set R of resources, and for which there is an edge between two resources r_i and r_j if there is at least a common tag assigned to both resources regardless of the user. Formally, $E' = \{(r_i, r_j)|\exists((u, t_m, r_i) \in A \wedge (u, t_n, r_j) \in A \wedge t_m = t_n)\}$[5]. We have chosen this graph because previous studies such as [11] showed that tags tend to converge around resources in folksonomies.

By browsing the graph G', we look for domain relevant resources and collect the tags used to annotate those resources as valid terms within the domain at hand. Our approach requires a set $S \subset R$ of user-specified seeds. A seed is a folksonomy resource considered highly pertinent to the domain. Thus we can compare resources in G' with a seed in terms of shared tags; tags of highly similar resources are added to the domain terminology.

In order to traverse the graph G' we use a breadth-first search (BFS) strategy and spreading activation [7], a technique that allows taking decisions over each node visited in the graph so that we can gather the tags associated with nodes which are relevant to the domain. Spreading activation is a graph search method initiated by a set of seed nodes weighted with an activation value. Each seed's activation value spreads through the adjacent nodes in the graph by means of an activation function. The spreading stops when a node activation value is below a specified threshold. When a node is activated more than once, *i.e.* it is reached by the spreading of different seeds, the node activation value can be added up.

The activation value for a node r_j is calculated by estimating the rate of shared tags with r_i, being r_i the previous activated node from which we reached r_j.

$$a'(r_j) = \frac{|\{t \in T|(u, r_j, t) \in A\} \cap \{t \in T|(u, r_i, t) \in A\}|}{|\{t \in T|(u, r_i, t) \in A\}|} \quad (1)$$

The activation function also depends on the activation value of the previously activated node r_i. Thus, $a(r_j) = a'(r_j) + a(r_i) * \lambda$ where $0 \leq \lambda \leq 1$ is a real number representing a decay factor. If $a(r_j)$ is greater than a threshold h, then it is marked as activated and the search continues; otherwise the search stops.

Once all the seeds are processed we we gather all distinct tags used to annotate the activated nodes and produce with them a list of domain terms.

3.2 Semantic Elicitation

We rely on general purpose knowledge bases published as linked data to identify the semantics of the terms extracted from the folksonomy. Linked data are published using RDF and typed links between data from different sources [4]. Throughout this paper we refer to DBpedia [5], OpenCyc, and UMBEL, which are general purpose knowledge bases published as linked data.

DBpedia[6] contains knowledge from Wikipedia for close to 3.5 million resources; 1.6 million resources are classified in a cross domain ontology containing 272 classes. OpenCyc is a general purpose knowledge base; it has nearly 500.000 concepts, around 15.000 types of relationships, and approximately 5 million facts relating these concepts. UMBEL is a vocabulary and reference concept ontology designed to help content to interoperate on the Web. This ontology has 28.000 concepts and 38 types of relationships. These datasets are interlinked among them. DBpedia resources, and classes are connected to OpenCyc concepts using *owl:sameAs*, and to UMBEL concepts using *umbel#correspondsTo*. Since these data are available in RDF we can query them using SPARQL.

In this process we ground the terms to DBpedia resources representing their intended meaning. We then tap into the DBpedia ontology and interconnected data sets (*i.e.,* OpenCyc and UMBEL) to i) identify which of the terms correspond to ontology classes, and ii) identify the relationships between those classes. When searching for relations between classes in the linked data set new classes can arise from those

[5]Note that annotations can be made by different users u, however, to keep the notation simple we don't show this in the formalization.

[6]see http://dbpedia.org/

```
PREFIX dbpr:<http://dbpedia.org/resource/>
PREFIX dbpo:<http://dbpedia.org/ontology/>

# Query 1.
SELECT ?resource WHERE {
 ?resource rdfs:label ?label .
 FILTER(?label="TagNormalizedVersion"@en) }

# Query 2.
SELECT ?main_resource WHERE {
  <resource> dbpo:wikiPageRedirects ?main_resource .}

# Query 3.
ASK {?disambiguation dbpo:wikiPageDisambiguates <main_resource> }

# Query 4.
SELECT ?candidate WHERE {
  <disambiguation> dbpo:wikiPageDisambiguates ?candidates.}
```

Listing 1: Queries for identifying DBpedia resources for terms.

relations expanding throughout intermediate classes. With the set of classes and relations we generate the domain ontology.

3.2.1 Grounding terms to DBpedia

For each term we identify in DBpedia the resource that better represents its intended meaning. To do so we follow a procedure previously introduced in [9] that we summarize in this section. This approach benefits of: i) redirections pages to deal with tags that represent synonyms or morphological variations of a concept, ii) a spelling service to attempt to fix misspelled tags, iii) disambiguation pages to identify and disambiguate ambiguous tags, and iv) the correspondence of DBpedia resources with Wikipedia articles to obtain textual descriptions of each resource which can be consume during the disambiguation process

Tags are written without any restriction by users, and thus several slightly modified tags can refer to the same concept. Therefore tags needs to be normalized so that different tags that indicate the same entity can be identified. We use DBpedia resource names as standard names of concepts to which tags can be transformed. First we modify the tags to turn them into the standard notation of DBpedia resource names, which is based on the Wikipedia title capitalization style[7]. Then we verify, using Query 1 in listing 1, if the tag normalized version is the label of a DBpedia resource. If the tag is not the label of DBpedia resource we pass it through a spelling service that suggest DBpedia resource names as spelling suggestions and try again the query. Next we check, using Query 2 in the same listing, if that resource redirects to a main resource. If this redirection triple does not exists, then we use the initial resource as the main resource. The main resource represents the most frequent meaning of the term defined by Wikipedia editors.

However, for ambiguous terms, the most frequent meaning is not necessarily the meaning of the tag given the context where it was used. Therefore first we need to establish whether the resource is ambiguous or not. We use Query 3 in listing in listing 1 to check if the main resource is disambiguated by another resource. This disambiguation resources are extracted from Wikipedia and contains a list of resources representing the candidate meanings that can be obtained with Query 4 in the same listing. If the resource is not related to a disambiguation resource, we consider that the term is not ambiguous and therefore we use this resource as the one representing the term meaning. On the other hand, if the resource is related to a disambiguation resource, then we have to select the proper sense among the candidates.

For instance, let us follow the process for the annotation BankLoan. We pose the SPARQL Query 1 (see listing 1) to search for a resource associated with the term *BankLoan*. This query does not return a resource so we pass the annotation to the spelling service that returns Bank_Loan as the suggested spelling. We execute again Query 1, and this time it returns the resource *dbpr:Bank_Loan*. Next we need to make sure this resource is a main concept and not a redirection resource. Thus we use Query 2 and it returns the resource *dbpr:Loan* which is actually the main concept since *Bank_loans* is just considered in Wikipedia as an alternative label to that concept. Then we verify if *dbpr:Loan* is an ambiguous resource. Query 3 returns true, meaning that the dbpr:Loan is related to a disambiguation resource *dbpr:Loan_(disambiguation)*[8]. Thus we consider dbpr:Loan as an ambiguous resource and retrieve the resources representing the candidate meanings using query 4. We obtain the following candidates: *dbpr:Loan*, *dbpr:Loan_(sports)*, *dbpr:-_Loanword*, etc.

To select the right candidate for a term we leverage the correspondence of DBpedia resources with Wikipedia articles to obtain textual descriptions of each resource. Thus, we calculate similarity between each resource and the term by comparing the resource textual description with the term context. The most similar resource is selected as the resource representing the term meaning. By context we mean other tags used together with the ambiguous term when annotating resources that have been activated in the spreading activation. In other words, the context of an ambiguous term t_i is the set $\{t \in T | (u, t, r_m) \in A \wedge (u, t_i, r_n) \in A \wedge r_m = r_n\}$.

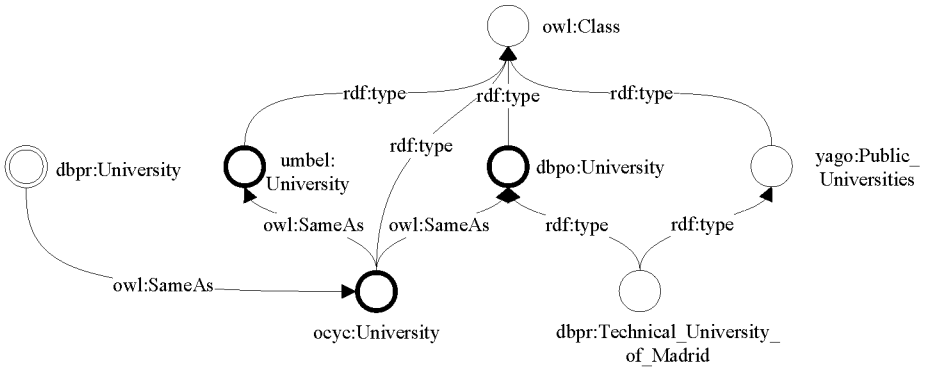
To calculate similarity between the term context and the resource description we use vectors to represent them and cosine as a measure of similarity. The components of the vectors are the most frequent terms of the Wikipedia articles related to each candidate DBpedia resource. To populate the vectors representing resources we use term frequency and inverse document frequency (TF-IDF) as term weighting scheme [22]. IDF is calculated using only the set of textual descriptions corresponding to the candidate resources. To populate the vector representing the term context we assign a weight of 1 if the vector term appears in the context, and 0 otherwise. Then we calculate the cosine of the angle between the vector representing the term with each of the vectors representing candidate resources. The candidate with the highest cosine is selected as the resource to represent the term. Details of this procedure can be found in [9]. This grounding produces a set of tuples relating terms with DBpedia resources from which we obtain a set of unique DBpedia resources.

3.2.2 Getting ontology classes

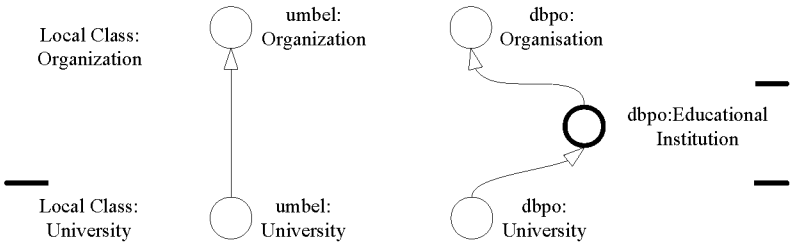
At this point we have connected terms to the linked data cloud using DBpedia resources. We aim at benefiting from

[7] http://en.wikipedia.org/wiki/Wikipedia:Article_titles

[8] For space reasons we do not include this query



(a) RDF graph of data linked to dbpr:University.



(b) Searching for relationships among grouped classes.

Figure 2: Browsing linked data for classes related to DBpedia resources and relations among classes.

```
# Query 1.
ASK{<resource> <rdf:type> <rdfs:Class>}

# Query 2
SELECT ?class
WHERE{ <resource> ?rel1 ?class. ?class <rdf:type> <rdfs:Class>
       FILTER (?rel1 = <owl:sameAs>) }

# Query 3
SELECT ?class
WHERE{ <resource> ?rel1 ?node. ?node ?rel2 ?class.
       ?class <rdf:type> <rdfs:Class>
       FILTER((?rel1 = <owl:sameAs>) &&
                       (?rel2 = <owl:sameAs>))}
```

Listing 2: Queries for identifying classes from DBpedia resources.

the DBpedia ontology as well as from linked ontologies to identify which of the DBpedia resources obtained in the previous activity correspond to ontology classes. The most simple case occurs when a term is grounded to a DBpedia resource which represents a class in the knowledge base. If the DBpedia resource represents a class then must there exists a RDF statement stating that fact. We can look for such rdf statement using Query 1 in listing 2, which returns true or false if the resource r is defined as a class or not.

However, there is the case when a DBpedia resource has not been defined as a class directly but through other linked entities. For instance, the resource *dbpr:University* is not directly related to the class *dbpo:University*[9] (see figure 2a). To understand this equivalence we ought to navigate through linked data sets following the next path: *dbpr:University owl:sameAs ocyc:University* [10] *owl:sameAs dbpo:University*. Furthermore in this subgraph we can identify two more University classes in OpenCyc and UMBEL which are linked to the initial DBpedia resource.

Therefore, to identify classes related to a DBpedia resource r we query, using SPARQL, the linked data set in order to find paths of *sameAs* relations and of variable length connecting r with a target entity c defined as a class. We follow a similar approach to [12] where queries are used to traverse all the possible paths in the RDF graph connecting two entities. We define the path length L as the number of relationships found in the path linking r with c. For $L = 1$ we look for a pattern containing a relationship $relation_i$ linking r with c. As we do not know the direction of $relation_i$, we search in both directions: 1) $r\ relation_i\ c$, and 2) $c\ relation_i\ r$. Query 2 in listing 2 present one of the two queries.

For $L = 2$ we look for a path containing two relationships and an intermediate resource *node* such as: $r\ relation_i\ node$, and $node\ relation_j\ c$. Note that each relationship may have two directions and hence the number of possible paths is $2^2 = 4$. Query 3 in listing 2 is one of the four queries posed for paths of length 2. For $L = 3$ we have two in-between nodes, three relationship placeholders, and the number of possible paths is $2^3 = 8$. In general, for a path length L we have $n = \sum_{l=1}^{L} 2^l$ possible paths that can be traversed by issuing the same number of SPARQL queries on the linked data set.

In short, for each DBpedia resource r and a given value of L we pose n SPARQL queries following the above pattern to find related classes. Please note that the use of property paths in SPARQL 1.1 would allow simplifying this process. As a result a set of tuples associating each DBpedia resource

[9]The *dbpo* prefix refers to `http://dbpedia.org/ontology/`

[10]The prefix *ocyc* refers to `http://sw.opencyc.org/2009/04/07/concept/en/`

with ontology classes *oc* are created. Since a resource can be related to more than one linked data class we create a unique class, called local class *lc*, to group the linked data classes that the resource is related to. Each local class *lc* is associated, by means of a *owl:sameAs* relationship, with the classes *oc* identified for the resource. Hence, in this task we create an ontology O and then add the local classes *lc* along with their relationships with the classes *oc* to the ontology. From the example, we add to the ontology a local class *University* and assert that it is *owl:sameAs* to *dbpo:University*, *ocyc:University* and *umbel:University*.

3.2.3 Finding relationships and new classes

To look for relationships among the local classes *lc* we use their grouped classes *oc* to search for relationships among them in the linked data set. We carry out a pairwise search for relationships among the *oc* classes of all local classes. In order to benefit the most from the linked data graph, we need to look for variable length paths of relationships. Thus, we follow a similar strategy to the one for finding classes for DBpedia resources. Nevertheless, in this case we have a concrete source oc_i and target oc_j of the path; given that both classes are related to different local classes lc_i and lc_j. Classes found in a path linking oc_i and oc_j are added as local classes to the ontology O. In addition, for each relationship found between oc_i and oc_j we create relationships between lc_i and lc_j.

Figure 2b shows existing relationships between grouped classes of two local classes: *University* and *Organization*. Looking for a path of length 1 between members of both local classes in the dataset we find: *umbel:University rdfs:subClassOf umbel:Organization*. Therefore we can assert that *University* is *rdfs:subClassOf* of *Organization*. For paths of length 2 we find that *dbpo:EducationalInstitution* sets a path between the classes *dbpo:University* and *dbpo:Organisation*. In this case we add to the ontology a new class *EducationalInstitution* along with its corresponding relationships.

Of course this strategy of collecting relationships and classes may result in adding non relevant domain information to the ontology if we set L to a large value or if the path between two classes passes through abstract or high level concepts. Regarding the path length we think that if we keep L small, meaning a short path length, we are still able to collect relevant relationships and classes while avoiding unwanted information. In fact, in our experiments, presented in section 4, we show satisfactory results with short values of L that allow us to enrich the initial set of classes while preserving the precision of the ontology.

High level concepts like *owl:thing*, *umbel:RefConcept*[11] and *cyc:ObjectType*[12], or broad relationships such as *skos:broaderTransitive*[13] are another threat even with a short path length since they potentially can connect many unrelated or loosely related classes. Nevertheless, these abstract concepts and broad relationships can be easily identified since their are part of the knowledge base core vocabulary which is usually well documented (e.g., UMBEL vocabulary[14], DBpedia Ontology[15], and Cyc vocabulary[16]). Thus, in our experiments we create lists of high level concepts collected from the knowledge base vocabularies to filter out the paths containing those concepts. More sophisticated solutions can be applied to this problem too. For instance we can measure the semantic distance between two classes by following Wu and Palmer approach [27] where they take into account variables such as the depth of the class in the taxonomy and the depth of the least common subsumer to both classes, or the work of Jiang and Conrath [16] where authors propose a semantic distance measure which includes the local network density (i.e., subclasses per class), class depth, weights according to link types, and the information content of the classes and the least common consumer.

In summary, the result of this process is an ontology created out of folksonomy terms and of the reuse of knowledge contained in existing ontologies. A initial set of ontology classes are identified in the linked data set from folksonomy terms, while some others are collected when searching for relations among the initial classes.

[11]http://umbel.org/umbel#RefConcept
[12]http://sw.opencyc.org/2012/05/10/concept/en/ObjectType
[13]http://www.w3.org/2004/02/skos/core#broaderTransitive
[14]http://umbel.org/specifications/vocabulary
[15]http://mappings.dbpedia.org/server/ontology/classes/
[16]http://www.cyc.com/kb/thing

4. AN EXPERIMENT IN THE FINANCIAL DOMAIN

In this section we evaluate different ontologies in the stock market domain that were generated from a folksonomy excerpt extracted from Delicious, and linked data from DBPedia, OpenCyc, and UMBEL. Domain experts provided relevant websites for the financial domain including sites of markets such as NYSE, and NASDAQ, and of information providers including Yahoo! finance, among others. We used these sites as seeds during the Terminology Extraction phase. Our Delicious dataset (*Delicious Popular URLs and Tags, version 1.0*) was comprised of 100,000 URLs; each URL had been saved at least 100 times. The ten most commonly used tags for each URL along with their annotation frequency were available.

4.0.4 Terminology extraction

We obtained the graph G' from the input folksonomy and then used the seeds as starting points for the spreading activation process. Since the terminology gathered by the spreading activation process depends on the threshold h, we tested our method with the following h values: 0.5, 0.6, 0.7, and 0.8. Recall that this threshold is used to decide whether the spreading continues or not. We defined 0.5 as the lowest value since it guarantees that at least half of the tags are shared between seeds and the resources. To evaluate the relevance of the terminology, we compared it to three financial glossaries: Yahoo! Financial Glossary (http://biz.yahoo.com/f/g/), Investor Words (http://www.investorwords.com/), and Campbell R. Harvey's Hypertextual Finance Glossary (http://www.duke.edu/~charvey/).

We measured the precision of our method as the fraction of terms collected by our approach which are found in each one of the glossaries. Results are shown in Table 1a. A lower threshold allows collecting more terms at the expenses of a lower precision, while a higher threshold assure more relation between the activated nodes in terms of the shared tags, and hence a smaller set of nodes are activated from which a more precise terminology is collected. A threshold of 0.8 produces a terminology with the highest average precision. We compared the terms against the Yahoo! glossary

Table 1: Evaluation results of the terminology and classes
(a) Precision of the terminology using different threshold values

		Precision			
Threshold	**Terms**	**Yahoo!**	**Investor W.**	**H. Campbell**	**Average**
0.5	476	46.01%	32.56%	14.71%	31.09%
0.6	220	48.18%	37.73%	16.36%	34.09%
0.7	114	55.26%	40.35%	17.54%	37.72%
0.8	58	**94.83%**	48.28%	25.86%	**57.32%**

(b) Precision and recall of the ontology classes for different thresholds

Threshold	**Onto. Classes**	**Recall**	**Recall***	**Precision**	**Precision***
0.5	243	36.21%	48.28%	5.35%	16.05%
0.6	204	36.21%	46.55%	5.88%	21.57%
0.7	101	36.21%	44.83%	10.89%	33.66%
0.8	45	32.76%	43.10%	20.00%	68.89%

(c) Ontology Modules and precision of the classes

Module	**Classes**		**Module**	**Classes**	
	% of Total	**Precision**		**% of Total**	**Precision**
Organization	30.48%	77.8%	Stock Exchange..	15.51%	84.6%
Company	13.90%	88.5%	Money Transaction	4.81%	100%
Person	4.81%	55.6%	Country	3.74%	100%
Union (Company)	3.74%	40%	Research	3.21%	100%
Banker	2.14%	100%	Driver	1.60%	0%
Human	1.07%	100%	Member	1.07%	100%

and reached a precision of 94.83%. The result we attained when comparing against H. Campbell and Investor's controlled vocabulary was lower; these thesauri often have terms composed of more than two words, for instance *Beggar-thy-neighbor devaluation* or *Depository Institutions Deregulation and Monetary Control Act*, as well as specialization of the terms, for instance rather than *Balance* we find *Balance on goods and services*. Such overly specialized terminology was not found to be common in folksonomies.

Please note that although this number of terms, and consequently the number of classes derived from them, may not seem significant, the fact that this information is being gathered automatically would allow ontology engineer teams to save time. Otherwise as stated in ontology development methodologies [25] engineers would need to work with domain experts to write competency questions from which they can draw an initial terminology. In fact, the number of terms gathered with the highest precision is not too far from the 70 classes in the OpenCyc finance vocabulary.

4.0.5 *Semantic elicitation*

For each terminology produced in the previous activity we generated an ontology that was evaluated by comparing it against an existing financial ontology. For this experiment we set *L=3* so that we could find relations spanning the three linked data sets.

To evaluate the ontologies we used as gold standard the OpenCyc financial and transfer of possession vocabularies. From each ontology generated with our method we selected only the classes identified from OpenCyc. With this subset of classes we calculated precision and recall. Precision is the fraction of classes in each ontology that are relevant. Relevant classes are those appearing in the gold standard. Recall is the fraction of relevant classes appearing in the generated ontologies. We also calculated precision* and recall* using another version of the gold standard where we included all the subclasses of the classes in the finance and transfer of possession vocabularies. Precision* was calculated as before while recall* is a relaxed version of the original metric. In recall* the set of relevant classes are those in the original gold standard though we consider that a relevant class appears in the generated ontology if we find the relevant class or one of its subclasses. Results are presented in table 1b.

These results show the trade-off between precision and recall, particularly between precision* and recall*. A low threshold h indicates that we impose fewer constraints to collect folksonomy terms and hence we have more classes in the ontology. This high number of classes results in a higher recall*, 48.28% for h=0.5, though the precision of these classes is lower. On the other hand, a high threshold h indicates that we enforce more relatedness between the nodes in the folksonomy to collect relevant terms and thus we have fewer but more related classes. This low number of classes yields a higher precision, 68.8% for h=0.8, though the recall is lower. As we will see in the following section recall values are affected since the produced ontologies focus more on the stock market domain, given the set of seeds utilized in our experiment, rather than on personal banking which is one of the components of the gold standard.

4.0.6 *Inspecting a financial ontology*

In this section we analyze in deep the semantics of an ontology generated using our method and present an evaluation, carried out by domain experts, of a domain ontology generated with our method. According to our experiments *h=0.8* produces the most precise terminology consisting of 58 terms. With this set of terms we executed the semantic elicitation activity. For this experiment we set *L=4* to

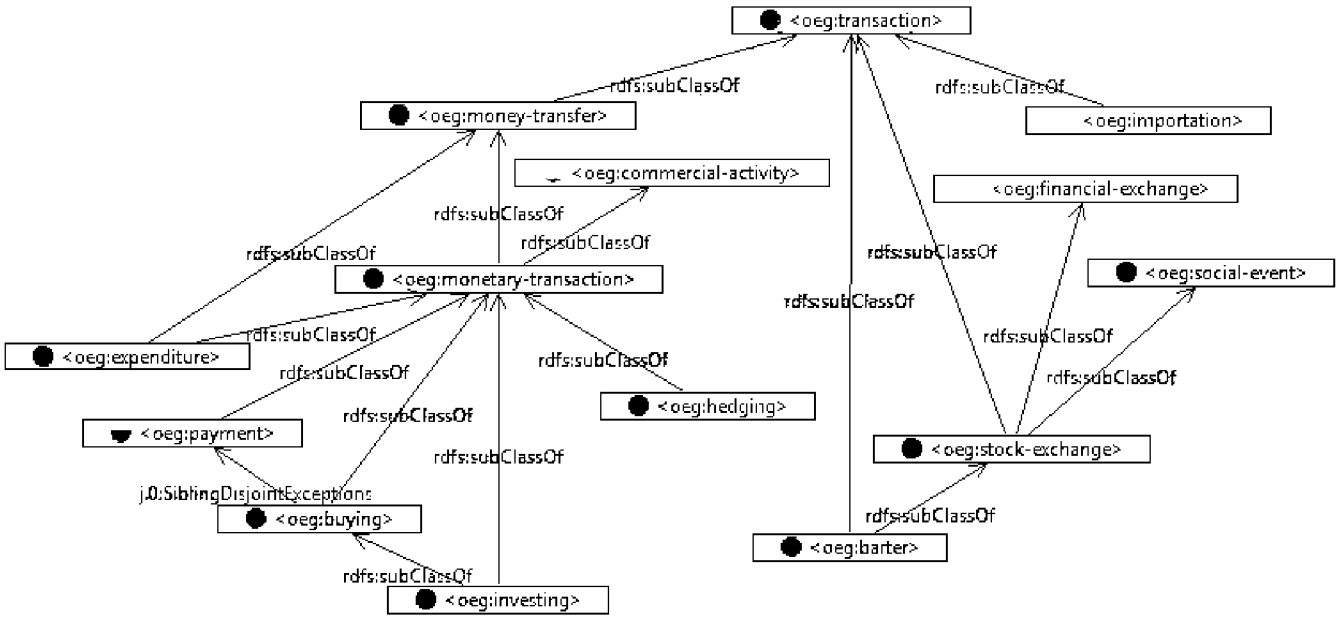


Figure 3: Excerpt from the financial ontology showing classes related to *Transaction*.

obtain more classes and relations. Our method was able to ground a total of 55 terms to DBpedia resources –94.83% corresponding to 42 different resources. The difference between the number of terms and that of DBpedia resources exposes that some terms refer to the same resource, yet they are different; this can be attributed to the use of synonyms or spelling variations. From those 42 resources, 23 correspond to at least one class in the linked data set. In total, 23 local classes were added to the ontology. We found 378 relationships and 164 new classes setting a path between initial classes.

The final ontology (see http://goo.gl/0v8zA) consists of 187 classes linked, by means of the *owl:sameAs* relationship, to 212 classes of three different ontologies in the linked data set. The ontology defines relationships corresponding to 8 different types including: *rdfs:subclassOf, owl:disjointWith, owl:sameAs, ocyc:SiblingDisjointExceptions*. Only ten classes were totally disconnected from the others. An excerpt of the produced ontology is shown in Figure 3.

To analyze the knowledge within the ontology, we decomposed it into modules using the partitioning tool PATO [23]. Modules generated by PATO are comprised of nodes for which the strength of the connection between the nodes inside the module is higher than the strength of any connection to nodes outside the module. A modular ontology facilitates analysis and maintenance operations. For our ontology, PATO identified eleven modules of interconnected nodes (see table 1c) and one module of isolated nodes. From these modules we observed that our ontology consists of a majority of topics that are relevant to the stock market. There are modules describing: 1) financial tasks such as Stock Exchange and Money Transactions, and 2) agents participating in those activities including Organizations, Companies, and Bankers.

To assess the suitability of the classes for the stock market domain we asked four domain experts to rate classes in each module according to their relatedness to the stock market domain. Precision was calculated as the fraction of relevant classes identified by our approach. A class was considered relevant if at least three evaluators asserted that it was highly related or related. Results of this evaluation are presented in table 1c. The overall precision of the ontology generated by our method was 80.67% (Evaluation data is available at: http://goo.gl/etztA). The results confirm that most of the ontology modules as well as the classes pertain to the domain. We obtained high precision values for Company, Stock Exchange, and Organization; we reached 100% precision for modules such as Money Transaction, Country, Research, and Banker.

For this evaluation Fleiss' Kappa [8] was κ=0.137; meaning that the evaluator's agreement exceeds the random result. Evaluators reached agreements by majority when rating 63.98% of the evaluated classes. However, evaluators were not consistent when rating. For instance, most of them chose the classes Organization and Company as relevant, although the same evaluators stated that some subclasses were not relevant. Only two modules, Union (Company) and Driver, had a precision lower than 50%, yet they only represent 3.2% of the classes.

We also evaluate the relationships between classes in the ontology. To do so we express them in natural language sentences using the SWAT tool (http://swat.open.ac.uk/tools/) so that evaluators were able to state the truth value of them. SWAT produces a set of sentences describing the ontology axioms. For instance, if we have an ontology asserting that Stock *rdfs:subClassOf* Equity and *owl:disjointWith* LoanNote, SWAT will produce two sentences: i) A Stock is an Equity and ii) No Stock is a LoanNote. Thus, we asked the evaluators to rate these sentences as true or false. Evaluators agreed in 70.33% of their ratings. Considering the sentences that evaluators were able to rate the 96.4% were valid relations.

5. CONCLUSIONS AND FINAL REMARKS

Folksonomies contain terms that can be harvested in knowledge acquisition processes, although these terms are not nor-

mally as specialized as those available in specialized glossaries. We have shown how those terms can be used for the generation of domain ontologies, by reusing classes in existing ontologies. Our method harnesses the DBpedia high degree of interconnection in the linked open data cloud, and explicit links between different data sets. According to the results obtained in the experiment we can claim that our approach helps in the development of ontologies that can be used as starting point for ontology engineers when developing a domain ontology. Our method produces a domain ontology that consists of classes and defined relationships between them, including hierarchical and non-hierarchical ones.

We must highlight that our method relies on the existing links within the linked data cloud, yet sometimes those links are inconsistent. For instance, there exists the class Stock Exchange in OpenCyc that is linked by a *owl:sameAs* relationship to the UMBEL class Exchange of User Rights. The problem arises because both classes are defined in different senses. The OpenCyc class refers to an organization that provides a marketplace where options can be traded, while the UMBEL class represents a transferring possession event. Thus, when using our method we may create ontologies with mistaken information, considering two classes that are different as equal.

Future work includes the use of more sophisticated techniques to evaluate the semantic distance between classes so that we can filter out non relevant information before adding new classes and relationships to the ontology. In addition we think that a natural way to improve this work is to benefit of the whole linked data cloud. Realizing this vision will require a technique for discovering data sets containing information relevant to the domain. Thus, we could query services such as CKAN (http://ckan.net/) for metadata about the data sets and semantic search engines, including Sindice (http://sindice.com/) and Watson (http://watson.kmi.open.ac.uk/). In addition, the identification of classes can be extended with services such as *sameAs.org*. Finally, once the datasets have been selected, we can use federated SPARQL queries to consume data stored across several SPARQL endpoints.

6. ACKNOWLEDGMENTS

This work has been funded by the EU FP7 project LIDER: "Linked Data as an enabler of cross-media and multilingual content analytics for enterprises across Europe" (610782).

7. REFERENCES

[1] A. Almeida, B. Sotomayor, J. Abaitua, and D. López-De-Ipiña. folk2onto: Bridging the gap between social tags and ontologies.

[2] S. Angeletou, M. Sabou, and E. Motta. Semantically enriching folksonomies with flor. In *In Proc of the 5th ESWC. workshop: Collective Intelligence amp; the Semantic Web*, 2008.

[3] G. Begelman, P. Keller, and F. Smadja. Automated tag clustering: Improving search and exploration in the tag space. *Collaborative Web Tagging Workshop at WWW2006 Edinburgh Scotland*, 22([2), 2006.

[4] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.

[5] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the Web of Data. *Journal of Web Semantic*, 7(3):154–165, 2009.

[6] C. Cattuto, D. Benz, A. Hotho, and G. Stumme. Semantic grounding of tag relatedness in social bookmarking systems. In *The Semantic Web - ISWC 2008*, volume 5318 of *Lecture Notes in Computer Science*, pages 615–631. Springer Berlin / Heidelberg, 2008.

[7] F. Crestani. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11:453–482, 1997.

[8] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378 – 382, 1971.

[9] A. García-Silva, I. Cantador, and O. Corcho. Enabling folksonomies for knowledge extraction: A semantic grounding approach. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 8:24–41, 2012.

[10] A. García-Silva, O. Corcho, H. Alani, and A. Gómez-Pérez. Review of the state of the art: discovering and associating semantics to tags in folksonomies. *The Knowledge Engineering Review*, 27:57–85, 2 2012.

[11] S. A. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2006.

[12] P. Heim, S. Lohmann, and T. Stegemann. Interactive relationship discovery via the semantic web. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *LNCS*, pages 303–317, Berlin/Heidelberg, 2010. Springer.

[13] P. Heymann and H. Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. *Stanford InfoLab Technical Report*, (2006-10):1–5, 2006.

[14] R. Jaschke, A. Hotho, C. Schmitz, B. Ganter, and G. Stumme. Discovering shared conceptualizations in folksonomies. *Web Semantics Science Services and Agents on the World Wide Web*, 6(1):38–53, 2008.

[15] Q. Ji, P. Haase, G. Qi, P. Hitzler, and S. Stadtmüller. RadonâĂŤrepair and diagnosis in ontology networks. In *The semantic web: research and applications*, pages 863–867. Springer, 2009.

[16] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, cmp-lg/9709008, 1997.

[17] C. Korner, D. Benz, M. Strohmaier, A. Hotho, and G. Stumme. Stop thinking, start tagging - tag semantics emerge from collaborative verbosity. In *Proceedings of the 19th International World Wide Web Conference (WWW 2010)*, Raleigh, NC, USA, Apr. 2010. ACM.

[18] K. Kotis and A. Vouros. Human-centered ontology engineering: The hcome methodology. *Knowledge and Information Systems*, 10(1):109–131, July 2006.

[19] C. Marlow, M. Naaman, D. Boyd, and M. Davis. Ht06, tagging paper, taxonomy, flickr, academic article, to read. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*,

pages 31–40, New York, NY, USA, 2006. ACM.

[20] P. Mika. Ontologies are us: A unified model of social networks and semantics. *Web Semant.*, 5:5–15, March 2007.

[21] H. Pinto, S. Staab, and C. Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, Valencia, Spain, 2004.

[22] G. Salton and M. J. Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[23] A. Schlicht and H. Stuckenschmidt. A flexible partitioning tool for large ontologies. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, pages 482–488, Washington, DC, USA, 2008. IEEE Computer Society.

[24] S. Sen, S. K. Lam, A. M. Rashid, D. Cosley, D. Frankowski, J. Osterhouse, F. M. Harper, and J. Riedl. tagging, communities, vocabulary, evolution. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 181–190, New York, NY, USA, 2006. ACM.

[25] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López. The neon methodology for ontology engineering. In M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, and A. Gangemi, editors, *Ontology Engineering in a Networked World*, pages 9–34. Springer Berlin Heidelberg, 2012.

[26] T. WeithÃűner, T. Liebig, M. Luther, S. BÃűhm, F. Henke, and O. Noppens. Real-world reasoning with owl. In E. Franconi, M. Kifer, and W. May, editors, *The Semantic Web: Research and Applications*, volume 4519 of *Lecture Notes in Computer Science*, pages 296–310. Springer Berlin Heidelberg, 2007.

[27] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *Proc. of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, 1994.