Departamento de Inteligencia Artificial
Facultad de Informática

PhD Thesis

A Method for Developing Ontologies from User-Generated Classification Systems

Author: Héctor Andrés García Silva
Supervisors: Dr. Oscar Corcho and Dr. Asunción Gómez Pérez

December, 2012

Tribunal nombrado por el Sr. Rector Magfco. de la Universidad Politécnica de Madrid, el día 23 de Octubre de 2012

Presidente:	D. PABLO CASTELLS AZPILICUETA
Vocal:	D. ENRICO MOTTA
Vocal:	D. EDUARDO MENA NIETO
Vocal:	D. JOSE MANUEL GOMEZ PEREZ
Secretario:	D. MARIANO FERNANDEZ LOPEZ
Suplente:	D. MIRIAM FERNANDEZ SANCHEZ
Suplente:	D. GUADALUPE AGUADO DE CEA

Realizado el acto de defensa y lectura de la Tesis el día 19 de Diciembre de 2012 en la Facultad de Informática

Calificacíon: ____________________

EL PRESIDENTE LOS VOCALES

EL SECRETARIO

Abstract

Web 2.0 applications enabled users to classify information resources using their own vocabularies. The bottom-up nature of these user-generated classification systems have turned them into interesting knowledge sources, since they provide a rich terminology generated by potentially large user communities. Previous research has shown that it is possible to elicit some emergent semantics from the aggregation of individual classifications in these systems. However the generation of ontologies from them is still an open research problem. In this thesis we address the problem of how to tap into user-generated classification systems for building domain ontologies.

Our objective is *to design a method to develop domain ontologies from user-generated classifications systems*. To do so, we rely on ontologies in the Web of Data to formalize the semantics of the knowledge collected from the classification system. Current ontology development methodologies have recognized the importance of reusing knowledge from existing resources. Thus, our work is framed within the NeOn methodology scenario for *building ontologies by reusing and reengineering non-ontological resources*. The main contributions of this work are:

- An integrated method to develop ontologies from user-generated classification systems. With this method we extract a domain terminology from the classification system and then we formalize the semantics of this terminology by reusing ontologies in the Web of Data.
- Identification and adaptation of existing techniques for implementing the activities in the method so that they can fulfill the requirements of each activity.
- A novel study about emerging semantics in user-generated lists.

Resumen

La web 2.0 permitió a los usuarios clasificar recursos de información usando su propio vocabulario. Estos sistemas de clasificación generados por usuarios son recursos interesantes para la extracción de conocimiento debido principalmente a que proveen una extensa terminología generada por grandes comunidades de usuarios. Se ha demostrado en investigaciones previas que es posible obtener una semántica emergente de estos sistemas. Sin embargo la generación de ontologías a partir de ellos es todavía un problema de investigación abierto. Esta tesis trata el problema de cómo aprovechar los sistemas de clasificación generados por usuarios en la construcción de ontologías de dominio.

Así el objetivo de la tesis es *diseñar un método para desarrollar ontologías de dominio a partir de sistemas de clasificación generados por usuarios.* El método propuesto reutiliza conceptualizaciones existentes en ontologías publicadas en la Web de Datos para formalizar la semántica del conocimiento que se extrae del sistema de clasificación. Por tanto, este trabajo está enmarcado dentro del escenario para *desarrollar ontologías mediante la reutilización y reingeniería de recursos no ontológicos* que se ha definido en la Metodología NeOn. Las principales contribuciones de este trabajo son:

- Un método integrado para desarrollar una ontología de dominio a partir de sistemas de clasificación generados por usuarios. En este método se extrae una terminología de dominio del sistema de clasificación y posteriormente se formaliza su semántica reutilizando ontologías en la Web de Datos.
- La identificación y adaptación de un conjunto de técnicas para implementar las actividades propuestas en el método de tal manera que puedan cumplir automáticamente los requerimientos de cada actividad.
- Un novedoso estudio acerca de la semántica emergente en las listas generadas por usuarios en la Web.

To my family for their support and love.

Agradecimientos

Agradezco enormemente a Asunción Gómez Pérez por la oportunidad que me brindó de ser parte del grupo de investigación y de llevar a cabo la tesis doctoral. Su voto de confianza y orientación han sido definitivos para llegar a buen término. También agradezco a Oscar Corcho por su ayuda y liderazgo durante el desarrollo de la tesis. Igualmente agradezco a Harith Alani, Chris Bizer, Kristina Lerman y a Daniele Turi por las colaboraciones realizadas mientras visitaba sus grupos de investigación.

Tengo mucho que agradecer a la gente que he conocido desde que inicié este viaje y que han hecho que todo el proceso haya sido más llevadero y divertido. Algunos se han ido y otros han llegado en medio del camino, así que espero no dejarme a nadie. Gracias a Mauro, Nicholas, Yadira, Ivonne, Sergio, Cesar, María C., Boris, Moisés, Luis J., Elena, María, Mari C., Jose, Luis, Víctor, Jorge, Mauricio, Bene, Manuel, Oscar, Esther L., Dani, Miguel Ángel, Esther N., Idafen, Raúl, Max, Pablo, Anja, Robert, Stefano, Jason, Roberto, John y Matheus. Este trabajo también va por los amigos de siempre y como ellos saben quiénes son no es necesario mencionarlos.

Y sobre todo agradezco a mi familia. A mi madre y a mi padre por ayudarme a ser todo lo que soy, y a mi hermano Diego por inspirarme a llegar cada vez más lejos. A Pablo, Milene, Made, Dani, Ana, Bryan, Pilar y Zeina por todo lo que hemos vivido en familia. A Mina por ser la mejor abuela del mundo. También agradezco a Sonia por estar junto a mí durante estos años, ayudándome con cada sonrisa suya a ver las cosas de otra manera, y por los momentos inolvidables que hemos compartido aquí y allá.

Finalmente agradecer a los proyectos GeoBuddies (TSI2007-65677-C02-01) y España Virtual (ALT0317) que sirvieron de marco para esta tesis, y a la beca FPI (BES-2008-007622) por la financiación durante el desarrollo de la tesis y las ayudas de movilidad (EEBB-I-12-05843, EEBB-2011-44146, SEST1000I001352XV0, BOE-A-2009-3947) para visitar centros en el extranjero.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

In recent years we have witnessed the transition from a Web where the content is generated mainly by the owners of websites to a more open and social Web where users are not only information consumers but also producers –*prosumers* (Tapscott and Williams, 2006). This new age of the Web, also known as Web 2.0[1], has brought a diversity of new social applications like *wikis*, *blogs*, *social networks*, *social bookmarks*, and *photo*, *music* and *video sharing sites.* These applications made it possible for Web users to contribute and share huge amounts of information.

Nevertheless the wealth of user-generated content poses some challenges and opportunities to improve the management and retrieval of this information. Web 2.0 applications have used different strategies to overcome this information overload problem, including the use of tags to annotate information resources, and the use of user-generated lists or collections to organize them. The innovation of these solutions was the user empowerment to organize their information according to their own vocabulary, as opposed to previous approaches where information was represented by keywords extracted from the resources (*i.e.*, bag of words model in traditional information retrieval approaches), or by metadata (*i.e.*, keywords or taxonomy categories) generated by groups of experts.

Tags serve multiple purposes in tagging systems, such as content organization, description, sharing and searching. In 2003, Delicious[2] was released as a social bookmarking tool where users were able to assign tags to *URLs* in a collaborative manner. One

[1]http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
[2]http://delicious.com/

year later, Flickr[1] was presented as a social network for photo sharing where users could assign tags to their own photos or to other photos from their colleagues. Nowadays, tagging is part of many popular applications such as *Amazon*, *YouTube* and *Last.Fm*, to name a few, where users can assign tags to products, videos and songs respectively. The success of tagging is attributed to two main factors (Hotho et al., 2006): (a) they are very easy to create so that users do not need any special skills or experience to tag, and (b) the benefits of tagging are immediate. In 2004, Vander Wal[2] coined the term **folksonomy** to describe the new structure of users, tags and objects. Folksonomy is defined as *the result of personal free tagging of information and objects (anything with a URL) for one's own retrieval. The tagging is done in a social environment (usually shared and open to others).*

More recently, Web 2.0 applications have started allowing users to classify information in lists. We call these lists **named lists** since users assign names to them so that they can be identified and shared with other users. For instance, Twitter[3], the microblogging platform, included a list feature in 2009 to organize other users in the platform. Similarly, Pinterest[4], a social photo sharing site launched in 2010, allows users to arrange photos in named lists, which are known as *boards*. Furthermore representative tagging systems such as Delicious and Flickr have also their own version of named lists. Delicious calls them *stacks* and encourages people to create them around a theme. In Flickr users can create lists, referred to as *groups*, which are created by users but are collaboratively maintained and updated. All these named list versions allow sharing the listed information with other users in the platforms.

From now on we will refer to folksonomies and named lists as **user-generated classification systems**, since the classification of information is carried out by individual users instead of groups of experts. Tags and named lists can be considered as category names under which resources are classified. These classification systems reflect the individual point of view of each user to organize information. In applications based on folksonomies or named lists other users may partially agree with the individual classifications by reusing existing tags to annotate a resource, or by subscribing to an existing list.

[1]http://www.flickr.com/
[2]http://www.vanderwal.net/folksonomy.html
[3]http://www.twitter.com/
[4]http://pinterest.com

1.1 User-generated classification systems and knowledge acquisition

Two characteristics have turned user-generated classification systems into interesting resources for **knowledge acquisition processes**: i) a large user community is classifying resources using their own vocabulary, and ii) other users of the platform can reinforce the individual classifications (*i.e.*, reusing existing tags or named lists) which can be interpreted as an agreement about the vocabulary used to classify resources. Gruber (2007) claimed that it is possible to observe an emergent semantics (Staab et al., 2002) from the aggregation of the individual annotations of shared resources in folksonomies. In fact, research about folksonomies has shown the emergence of vocabularies, which tend to stabilize over time around resources (Golder and Huberman, 2006) and users (Marlow et al., 2006). These vocabularies are elicited from the aggregation of the individual classifications of resources from which it is possible to identify the most used category names to organize them. In addition, as a consequence of the large user communities maintaining the folksonomies, they can be considered as a good sources of terminology frequently updated. This contrasts with controlled vocabularies, such as thesauri and taxonomies, which are generally created and maintained by groups of experts. Hence, one of the advantages of these systems is their ability to rapidly adapt to new changes in terminologies and domains. These findings led to research works such as (Specia and Motta, 2007) and (Mika, 2007) to claim that it was possible to use folksonomies in knowledge acquisition processes at large scale. In contrast, named lists have not been studied yet as source of knowledge.

As explained before users are not enforced to use a controlled vocabulary. They can use acronyms, spellings variations, and synonyms of a given concept as category names. However **user-generated classification systems lack semantics** (Angeletou et al., 2008; Cantador et al., 2008; Tesconi et al., 2008), and thus it is not possible to identify these relations between terms. Therefore, to leverage user-generated classification systems as knowledge sources the semantics of the terms have to be specified. Synonyms, acronyms, and spelling variations of a given concept must be identified so that they can be properly represented in the knowledge base, thus avoiding duplicity of information. Ambiguous tags have to be disambiguated so that they can be added to the knowledge base according to their intended meaning.

Some proposals (Begelman et al., 2006; Giannakidou et al., 2008; Jäschke et al., 2008; Mika, 2007) tackle the lack of semantics associated with category names by clustering them, in the hope that such grouping exposes their meaning. The clusters are created according to a defined relation among category names, usually relying on a definition of similarity (Cattuto et al., 2008; Markines et al., 2009). On the other hand, other authors (Angeletou et al., 2008; Cantador et al., 2008; Tesconi et al., 2008) address this problem by relating category names to semantic entities in ontologies. Clustering-based approaches have the drawback that the meaning of the relations grouping the category names is not explicitly identified, what hampers the incorporation of the clusters into a knowledge base. Ontology-based approaches depend strongly on the ontology coverage of the terminology in the user-generated classification system. A low terminology coverage limits the amount of knowledge which can be added to the knowledge base. Although some of the works produce an ontology, none of them have limited the scope of the output ontology to a given domain.

The objective of this thesis is **to design a method and supporting techniques to develop domain ontologies from user-generated classification systems** in the Web. Therefore our work can be classified as an ontology learning approach (Buitelaar et al., 2005; Gómez-Pérez and Manzano-Macho, 2004; Maedche and Staab, 2001). Ontology learning aims at the creation of ontologies from unstructured (e.g., text files or Web pages) or structured data (e.g., XML files or relational databases).

Recent methodologies for developing ontologies (Suárez-Figueroa et al., 2012) have recognized that reusing existing ontologies and external knowledge sources is important in order to benefit from existing conceptualizations, and to speedup the development process which potentially may save cost. In fact, the NeOn methodology (Suárez-Figueroa et al., 2012) defines scenarios to build ontologies by *reusing and re-engineering ontological and non-ontological resources.* Ontologies to be reused can be found on the web as independent ontologies, for instance by using search engines such as Watson[1] or Swoogle[2], or as part of a *linked data* set (Bizer et al., 2009a). Linked data are data sets published in RDF and connected among them (Berners-Lee, 2006). At the time of writing this dissertation there are 295 datasets in the Linked Open Data cloud

[1]Watson: http://kmi-web05.open.ac.uk/WatsonWUI/

[2]Swoogle: http://swoogle.umbc.edu/

comprising 31,634,213,770 triples and 503,998,829 links[1]. This network of linked data is also known as The Web of data (Bizer et al., 2009a). Among these data sets there are general purpose ontologies such as DBpedia (Bizer et al., 2009b) and OpenCyc[2] as well as domain ontologies such as GeneOntology [3] and GeoLinkedData[4].

In this thesis, we propose **an integrated method to develop domain ontologies from user-generated classification systems relying on existing ontologies published on the web of data**. This method taps into user-generated classification systems to obtain a vocabulary (*i.e.*, a list of terms) relevant to the domain. Then we identify the terms that can be considered as classes. To do so we reuse ontologies in the web of data so that we can figure out whether a given term leads to the identification of a class. In addition, we search in the ontologies that we are reusing for relations among the discovered classes. Thus the output is an ontology comprising classes and their relations within a given domain.

We have modeled our method for developing ontologies from user-generated classification systems as a workflow, where processes are arranged in an ordered sequence. The method consists of two main processes: *Terminology Extraction* and *Semantic Elicitation*. In the terminology extraction process the data are extracted from the user-generated classification system and pre-processed, so that they are cleaned and normalized. Then this data set is traversed to select the relevant terms in the domain. In the semantic elicitation process terms are grounded to semantic entities in a general-purpose ontology (*e.g.*, DBpedia ontology). By grounding we mean to associate terms with ontology entities representing the term intended meaning. Then from these semantic entities we identify which of them correspond to classes in the ontologies that we are reusing. Next we search in these ontologies for relations among the identified classes.

We use DISPEL (*Data-Intensive Systems Process Engineering Language*) (Martin and Yaikhom, 2011) to describe the workflow. This language is suitable to describe the method since it was designed to focus on the description and understandability of the process rather than in the implementation. The benefits of using DISPEL include i) a formal description of how the data is streamed and transformed within the workflow,

[1]see Linked Open Data cloud statistics at http://www4.wiwiss.fu-berlin.de/lodcloud/state/
[2]see http://sw.opencyc.org/
[3]see http://www.geneontology.org/
[4]see http://geo.linkeddata.es/

and ii) the possibility of adding semantic descriptions (*i.e.*, semantic entities) to the data.

To automatically carry out the processes identified in the method we identify existing techniques and adapt them to carry out each process. In the case of terminology extraction, we use *spreading activation* (Crestani, 1997) to collect the relevant domain terms. For the semantic elicitation we use the *vector space model* (Salton and Mcgill, 1986) to ground the terms to semantic entities, and SPARQL (Prud'hommeaux and Seaborne, 2008) queries to identify classes from the semantic entities, and to search for relations among the identified classes.

We carried out different experiments to test how our method helps to achieve our objectives. One of our main concerns was the reproducibility of these experiments. Hence we made public the evaluated data set and used standard evaluation metrics, whenever it was possible, such as precision and recall (Baeza-Yates and Ribeiro-Neto, 2011). For those experiments involving human evaluations we assessed their validity using agreement measures such as the kappa statistical measure (Fleiss, 1971). In this thesis we present experiments regarding individual activities of each of the main processes such as the normalization and semantic grounding of category names. We also include an experiment of the whole approach where we used data from an existing user-generated classification system to produce a domain ontology. In addition, we include a novel experimental survey of the emergent semantics which can be elicited from named lists, which according to our review or the state of the art had not been studied yet.

1.2 Thesis structure

In the following we present the thesis structure:

Chapter 2. State of the Art. We review the state of the art regarding the use of folksonomies and named lists as data sources for knowledge acquisition processes. From this survey we identified the two open research problems addressed in this thesis: i) The lack of an integrated method and a set of supporting techniques to automatically elicit formal domain ontologies from user-generated classification systems, and ii) The lack of a survey about the emergent semantics in named lists. In addition we present a review of the methodologies for building ontologies where we show how our method solves problems not addressed by current methodologies.

Chapter 3. Objectives and Contributions. We present the global and specific objectives of this thesis work which were defined according to the open research problems identified in chapter 2. We also present the contributions to the state of the art, the assumptions and hypotheses on which our contributions rely, and the restrictions which define the scope of the different contributions. In addition we describe the research methodology that we followed during the development of this thesis.

Chapter 4. Method for Developing Ontologies. We describe our method to obtain ontologies from user-generated classification systems. To do so, we define processes, activities and tasks which have to be carried out, and the user roles involved in their execution. We use an example to illustrate each part of the method. The method consists of the following processes:

- **Terminology extraction**. We propose to extract the data from the user-generated classification systems and normalize them. Then from these data set we propose to select a sub set of relevant domain terms.
- **Semantic Elicitation**. In this process we aim at identifying the semantics of the relevant domain terms identified in the previous process. To do so we propose to reuse existing conceptualizations in knowledge bases. First we ground the set of terms to semantic entities that represent their meaning. Next we identify, in the knowledge base, classes from the semantic entities. Finally we search, in the knowledge base, for relations between pairs of classes.

Chapter 5. Techniques Supporting the Method. For each of the processes proposed in the method we provide techniques to achieve their goals. These are existing techniques which have been adapted to the particularities of the information in user-generated classification systems.

- We use spreading activation (Crestani, 1997) for gathering a domain relevant terminology from the user-generated classification system.
- We turn the semantic grounding of category names into a search task. That is, for a category name we search, among a set of candidate ontology entities,

the entity that better represents its meaning. To carry out this search we use the vector space model (Salton and Mcgill, 1986).

- To identify classes from the semantic entities, to which the category names have been grounded, we pose SPARQL queries over the ontologies that we are reusing in our process.
- Similarly to search for relations between the identified classes we pose SPARQL queries over the ontologies that we are reusing in our process.

Chapter 6. Evaluation. To test our method and supporting techniques we present experiments with regard to the normalization and the semantic grounding of category names. We also include an experiment covering all the processes proposed in the method where we produce a domain ontology from a subset of a real user-generated classification systems. Finally, in this section we include a novel survey about the emerging semantics in named lists.

Chapter 7. Conclusions and future work. We finalize this thesis presenting the conclusions of our research. We include potential benefits of using ontologies in the user-generated classification systems from which they were obtained. We also present the future work.

Annex A. Ontology for describing the method data. In this annex we present the ontology defining the semantics of the data structures used in the definition of each one of the method components. In chapter 6 each method component is described in terms of their input and output data using DISPEL. This language allows enriching the data definitions with semantic information provided by an ontology.

1.3 Dissemination of results

The contributions produced within the framework of this thesis have been published in international peer-reviewed journals, conferences and workshops. Some contributions are still in reviewing process so they are marked accordingly. In the following we list the contributions along with the publications which support them.

- A survey of the state of the art regarding the use of folksonomies as source for knowledge acquisition processes.

  García-Silva, A., Corcho, Óscar, Alani, H. and Gómez-Pérez, A. (2012) **Review of the state of the art: Discovering and Associating Semantics to Tags in Folksonomies**. The Knowledge Engineering Review, 27 (01). pp. 57-85. ISSN 0269-8889.

  García Castro, Raul and García-Silva, A. (2009) **Content Annotation in the Future Web**. The European Journal for the Informatics Professional, X (1). 27 - 32. ISSN 1684-5285

  García-Castro, R. and García-Silva, A. (2009) **Anotación de contenidos en la Web del futuro**. Novática (197). pp. 28-32. ISSN 0211-2124

- A process for the semantic grounding of tags in Folksonomies. The semantic grounding is carried out by associating tags with semantic entities in ontologies.

  García-Silva, A., Cantador, Iván, and Corcho, Óscar (2012) **Enabling Folksonomies for Knowledge Extraction: A Semantic Grounding Approach**. International Journal on Semantic Web and Information Systems, Special Issue on Web-Scale Knowledge Extraction, 8 (3).

  García-Silva, A., Szomszor, M., Alani, H. and Corcho, Óscar (2009) **Preliminary Results in Tag Disambiguation using DBpedia**. In: First International Workshop Collective Knowledge Capturing and Representation CKCaR09, September 2009, Redondo Beach, California, USA.

  García-Silva, A., Corcho, Óscar and Gracia, J. (2010) **Associating Semantics to Multilingual Tags in Folksonomies (Poster)**. In: 17th International Conference on Knowledge Engineering and Knowledge Management, EKAW2010, 11/10/2010 - 15/10/2010, Lisboa, Portugal.

- A process for the automatic generation of domain ontologies from folksonomies.

  In revision: García-Silva, A., García-Castro, Jael, García, A. and Corcho, Óscar and Gómez-Pérez, Asunción (2011) **Folksonomies and Linked Data for Ontology Development: A Case Study in the Financial Domain**. IEEE Internet Computing.

- A survey about the semantics which can be obtained from user-generated lists.

  García-Silva, A., Kang, Jeon-Hyung, Lerman, Kristina, and Corcho, Óscar (2012) **Characterising Emergent Semantics in Twitter Lists**. In: 9th Extended Semantic Web Conference, ESWC12, 27/05/2012-31/05/2012, Heraklion, Crete, Greece.

In addition, in the exploratory research stage of this thesis work, while defining the main research problem, we experimented with the possible uses of some of the techniques developed for the semantic grounding. Though the publications of these experiments are not strictly related with the contributions of this thesis, we consider them highly related to the thesis subject since they exploit some of the thesis findings in other related contexts.

- We used the disambiguation algorithm proposed in the semantic grounding process for the automatic annotation of natural language text.

  Mendes, P., Jakob, M., García-Silva, A. and Bizer, C. (2011) **DBpedia Spotlight: Shedding Light on the Web of Documents**. In: 7th International Conference on Semantic Systems, 7-9 September 2011, Graz, Austria.

  Muñoz-García, O., García-Silva, A., Corcho, Óscar, de la Higuera Hernández, M. and Navarro, C. (2011) **Identifying Topics in Social Media Posts using DBpedia**. In: Networked and Electronic Media Summit (NEM summit 2011), 27-29 September 2011, Torino, Italy.

- We adapt the semantic grounding for an approach to enrich ontologies with tagged multimedia information.

  García-Silva, A., Jakob, M., Mendes, PN. and Bizer, C. (2011) **Multipedia: Enriching DBpedia with Multimedia Information**. In: The Sixth International Conference on Knowledge Capture, K-CAP 2011, 25/06/2011-29/06/2011, Banff, Alberta, Canada.

1.4 Collaborations

Within the framework of this thesis the Ph.D student visited different research groups that were working in a highly related subject to that of his thesis. The goals of these visits were to explore new research opportunities by identifying intersection points between his work and the host group work, and to find new use cases where to apply the contributions produced in the thesis. In the following we list these research visits and mention the papers that were produced during each visit. Please note that the details of each paper were already provided in the previous section (1.3 Dissemination of Results) and thus we only include the paper title.

- **University of Southampton** (The United Kingdom), School of Electronic and Computer Science, Intelligence, Agents and Multimedia group. Year 2009. Length: 3 months. Host Researcher: Dr. Harith Alani. The outcome of this visit were two papers:
  - **Review of the state of the art: Discovering and Associating Semantics to Tags in Folksonomies.**
  - **Preliminary Results in Tag Disambiguation using DBpedia.**
- **Free University of Berlin** (Germany), School of Business and Economics, Web-based Systems Group. Year 2010. Length: 6 months. Host Researcher: Prof. Dr. Christian Bizer. The outcome of this visit were two papers:
  - **DBpedia Spotlight: Shedding Light on the Web of Documents.**
  - **Multipedia: Enriching DBpedia with Multimedia Information.**
- **University of Southern California** (The United States), Information Science Institute, Information Integration Research Group. Year 2011. Length: 4 months. Host Researcher: Dr. Kristina Lerman. The outcome of this visit was one paper:
  - **Characterising Emergent Semantics in Twitter Lists.**
- **Time Out** (The United Kingdom), Research and Personalisation. Year 2012. Length: 2 months. Host Researcher: Dr. Daniele Turi. At the time of writing this thesis we are writing a paper about the use of topics to model tweets in the entertainment domain.

CHAPTER 2

STATE OF THE ART

With the advent of the Web 2.0, users were allowed to create content that needed to be organized so that it could be successfully retrieved and shared with other users. Therefore, Web applications provided functionalities that allow users to manage their own information. Two representative functionalities to manage information in the Web are folksonomies and the use of named lists. Though these user-generated classification systems have features in common, folksonomies have been studied in depth by researchers while named lists have been unnoticed for them so far. A possible reason for this difference is that folksonomy-based systems, which appeared in 2003, gained a lot of attention due to their success for finding things with an strategy based on the exploration and serendipity[1]. In contrast named lists reflect traditional directories which were thought not apt to classify the large amount of content in the Web, mainly because of the limitations imposed by a small group of maintainers. According to this distinction in the amount of research efforts received by folksonomies and named lists we have decided to split this state of the art chapter in two sections: i) a brief section to describe named lists in the current web (see section 2.2) and ii) a section where we review the research about folksonomies as source of knowledge (see section 2.3). We finalize each section by identifying the open research problems regarding the use of these user-generated classification systems in the ontology development process.

Our focus in this thesis is to leverage the emergent semantics from user-generated classification systems for building ontologies. Thus, in section 2.1, we discuss the mean-

[1]see `http://vanderwal.net/folksonomy.html`

ing of the concept of emergent semantics in our context. We also included, in section 2.4, a review of the main methodologies for building ontologies where we show that none of them provide methods and techniques for building ontologies from user-generated classification systems.

2.1 Emergent semantics

In a general context emergence is defined as the arising of novel and coherent structures, patterns and properties during the process of self-organization in complex systems (Goldstein, 1999). Emergence phenomena can appear in distinct types of systems including physical systems or computer simulations. This concept is interesting in sciences since it allows explaining complex systems from the configuration of the components in contrast to use explanations based on the parts alone.

In computer science the term emergent computing refers to highly complex processes arising from the cooperation of many simple processes (Ruskin and Walshe, 2006), rather than from a designer's elaborate plan (Staab et al., 2002). These complex processes are characterized by systematic properties which are those that the system has but none of its parts have (Brunner, 2002).

In the context of artificial intelligence and the semantic web emergent semantics has been referred as to let semantics emerge out of the interaction of human and software agents (Staab et al., 2002). The concept has been also discussed within the context of semantic interoperability which is the problem of how to provide transparent access to heterogeneous information sources (Aberer et al., 2004). The benefits of emergent semantics have been recognized by the Semantic Web community since this observable semantics can be useful to ameliorate the knowledge acquisition bottleneck (Staab et al., 2002).

Folksonomies are an example of a system from which it is possible to observe an emergent semantics (Gruber, 2007). Individual annotations are important for the user itself though the aggregation of these annotations turn them in a useful data source in general. Golder and Huberman (2006) and Marlow et al. (2006) showed that it is possible to identify an emergent vocabulary around resources and around users in folksonomies. These findings led to a large number of research works to focus on the problems of: i) how to obtain the emergent semantics from folksonomies, and ii) how to develop ontologies out of that emergent semantics. In section 2.3 we present the most important research works addressing these problems. In this thesis we demonstrate that lists created by users in the Web are also a source of emergent semantics. In section 2.2 we present the evolution of lists in the Web from traditional web directories to the lists created by users.

2.2 Named lists

Named lists in the Web can be traced back to early 90s when the World Wide Web was created. Yahoo! first product, launched in 1994, was a human-edited directory[1] where web sites where organized in a hierarchy of categories and subcategories. A more sophisticated example is the Open Directory project [2] which was launched in 1998. The Open Directory is a multilingual web directory which manages more than 80 languages. The hierarchy of categories and web sites is maintained by a global community of volunteer editors. As of April 2012 the Open Directory has around 5,018,891 sites, 95,016 editors, and over 1,010,596 categories. Naturally the large taxonomies agreed by editors in the Yahoo directory and the Open directory have been recognized as lightweight ontologies which can be leveraged in distinct applications such as the definition of user profiles (Liu and Maes, 2004; Sieg et al., 2010) and ontology learning methods and tools (Kavalec and Svátek, 2002; Labrou and Finin, 1999; Varma, 2002).

2.2.1 Named lists in the Web 2.0

More recently, Web 2.0 applications have started allowing users to classify information in lists. For instance, Twitter[3] included a list feature in 2009 as an answer for those users requesting better ways of organizing information in the platform[4]. Any user can create a list to organize other users so that he can see all the messages of the listed people at once. Other users can subscribe to public lists. Similarly to Twitter, Pinterest[5], a social photo sharing site launched in 2010, allows users to arrange photos in named lists

[1]see `http://dir.yahoo.com/`
[2]see `http://www.dmoz.org`
[3]see `http://www.twitter.com/`
[4]as reported in Twitter blog: `http://blog.twitter.com/2009/09/soon-to-launch-lists.html`
[5]see `http://pinterest.com`

which are known as *boards*. In this social site other users can follow public boards so that they receive updates of new content in those boards.

Representative tagging systems such as Delicious and Flickr have also their own version of named lists. Delicious calls them *stacks* and encourages people to create them around a theme. Other users can follow public stacks so that they can see the listed information and receives updates. On the other hand, in Flickr users can create lists which are referred as *groups*, *sets* and *collections* to arrange their pictures. Groups are created by a user but are collaboratively maintained and updated. Users can join groups so they can see and receive updates of the information of the group. Sets and collections are also created by users but they are used to share information out of the platform.

2.2.2 Emergent semantics in Twitter lists

In this section we aim at illustrating that it is possible to identify an emergent semantics from named lists in Twitter. As it was mentioned before users in Twitter are allowed to classify other users into lists. The creator of the list is known as the curator. List names are freely chosen by the curator and consist of keywords. Users other than the curator can then subscribe to receive tweets from the listed users.

We can analyze term co-occurrence patterns in these lists to identify semantic relations between all these elements. Co-occurrence may happen due to the simultaneous use of keywords in: i) different lists created by the same curator, ii) in lists followed by the same subscriber, or iii) in lists under which the same user is listed. Figure 2.1 depicts an example of lists and user roles. We can identify two candidate relations according to the simultaneous use of keywords by curators: *Semantic Web* and *Open Data*, and *SemWeb* and *Social Web*. Subscribers use of lists has set up a relation between *Open Data* and *Social Web*. In addition, taking into account users listed under different lists we can see that *Semantic Web* and *Social Web*, and *Semantic Web* and *SemWeb* are related. In this case the latter relation is reinforced since two users are listed under both lists.

Another example is presented in table 2.1 where we summarize the lists under which an active and well known researcher in the Semantic Web field has been listed. The first column presents the most frequent keywords used by curators of these lists, while the second column shows keywords according to the number of subscribers. We can see

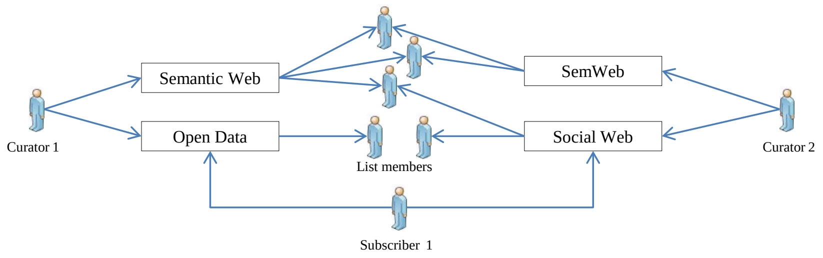


Figure 2.1: Diagram showing different user roles in twitter lists. Boxes indicate list names.

that *semantic_web* and *semweb* are frequently used to classify this user, which suggests a strong relationship between both keywords. In fact, these keywords can be considered as synonyms since they refer to same concept. Though less frequent, other keywords such as *semantic*, *tech* and *web_science* are also related to this context. The other keywords according to the use given by subscribers (*e.g.*, *connections*) are more general and less informative for our purposes.

Table 2.1: Most frequent keywords found in list names where the user has been listed.

Curators		**Subscribers**	
semantic_web	39	semantic_web	570
semweb	22	semweb	100
semantic	7	who-my-friends-talk-to	93
tech	7	connections	82
web_science	5	rock_stars	55

2.2.3 Conclusions

In this thesis, we distinguish between collaborative or proprietary efforts to create a unique hierarchy, such as in the case of the Web directories, from those where the classification system emerges from the individual classifications of users, which is the case of the lists in the Web 2.0 applications. We are interested in studying these emerging classification systems as potential sources of knowledge that can be leveraged for developing ontologies. Named lists and folksonomies are similar since in both classification systems users classify resources under a category name which can be a tag or a list name. In spite of this similarity named lists have not been widely researched as a potential source of emergent semantics.

Thus we can conclude that named lists constitute an interesting emerging classification system where a potentially large community of users classify resources under lists which are identified by names assigned freely by users. In the review of the state of the art we did not find research works aiming at tapping into these classification systems with the objective of identifying the semantics that can emerge from the different connections between list names with users and resources. Therefore we can state the following open research problem:

- Is it possible to elicit knowledge, in the form of semantically related terms and explicit relations between them, from the classification systems emerging from named lists?

2.3 Folksonomies as source of knowledge

Our goal is to survey research works studying Folksonomies as a source of knowledge, which can be later used to create ontologies. To do so first, in section 2.3.1, we identify the weakness of folksonomies from a semantic point of view. Then we present, in section 2.3.2, a unified process that we use to describe in a uniform way the approaches aiming at eliciting knowledge from folksonomies. We also describe a simple folksonomy in section 2.3.3, which we use to exemplify the results obtained from each one of the analyzed approaches. Please note that we did not have access to the programs implementing the surveyed approaches, and thus the results of applying a given approach to the example folksonomie were obtained by a dry run process[1]. In section 2.3.4 we present research works surveying different types of tag relatedness measures which are the foundation of some of the approaches for acquiring knowledge from folksonomies. Next we describe the approaches to enrich or discover the semantics in folksonomies. We present them according to their types: statistical-based (see 2.3.5), ontology-based (see 2.3.6), and hybrid (see 2.3.7). We also present a summary of the review in section 2.3.8. Finally, we present our conclusions.

[1]Dry run testing: http://en.wikipedia.org/wiki/Dry_run_%28testing%29

2.3.1 Weakness from the semantics point of view

We start by defining the two problems, from the semantic point of view, that affect current tagging technology. First problem is that **folksonomies lack a uniform representation to facilitate their sharing and reuse**. Some Web 2.0 applications provide APIs to export their folksonomies. However, they do it in proprietary formats. To overcome this problem, ontologies have been proposed to model the tagging activities in folksonomies, with semantic concepts to represent users, tags, resources, etc. (Echarte et al., 2007; Gruber, 2007; Kim et al., 2008a; Knerr, 2006; Newman, 2005; Passant and Laublet, 2008; Scerri et al., 2007). One example of these ontologies is the SCOT ontology (Kim et al., 2008a), which is depicted in Figure 2.2. This ontology models tagging information, and includes concepts such as *User*, *Item*, *Tag* and *Tag Cloud* as well as the relationships among these concepts. SCOT reuses existing vocabularies such as FOAF[1] and SIOC[2], being the former a set of classes and properties describing people and their interests, and the latter a popular ontology for interlinking online communities (Breslin et al., 2006). Some surveys in this respect have been published such as Kim et al. (2008b) where authors review most of the current ontologies for folksonomy information representation. Note that this lack of uniform representation problem as well as the proposed solutions are more related to data interchange than to the knowledge that can be extracted from folksonomies, and hence we consider this issue as out of the scope of our survey.

The second and more relevant problem is **the lack of formal and explicit semantic of tags, which hampers the reuse of the knowledge hidden in folksonomies**. This problem has been widely reported in Angeletou et al. (2008); Golder and Huberman (2006); Lee and Yong (2007); Szomszor et al. (2008). Users can use different morphological variations of a tag to represent the same label such as plurals, acronyms, conjugated verbs or misspelling words (e.g., different users can annotate a picture of a celebration with tags such as *party*, *parties*, *partying*, *partyign*). Furthermore, a user can use a tag to annotate a resource while another user can use a synonym of that tag to annotate another resource (e.g., synonyms as *party* and *celebration*). Moreover, some tags can be polysemous, where the same word has more than one meaning, such as *party* as a celebration as opposed to *party* as a political organization. Most current

[1] `http://www.foaf-project.org/`
[2] `http://sioc-project.org/`

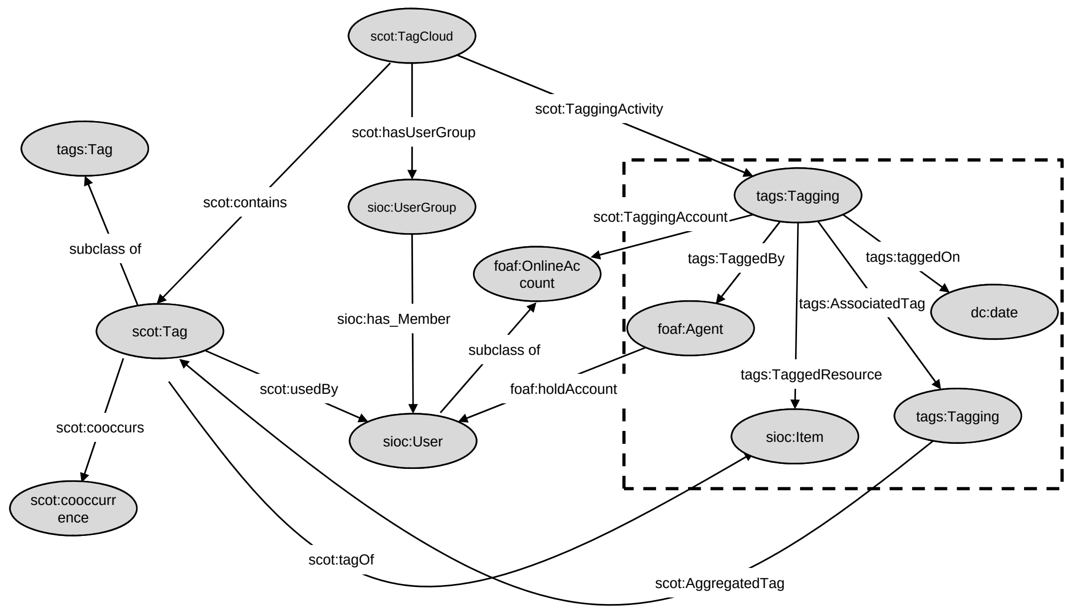


Figure 2.2: Graphical representation of the SCOT ontology (Kim et al., 2008a)

tagging applications do not allow users to define the intended meaning for their tags. For example, while Flickr tackles the ambiguity problem by providing clusters of related tags, the meaning of these tags, the meaning of their relationships, and the meaning of the cluster itself are not defined. Finally, different levels of granularity may be found in tags provided by users: some users issue more generic tags while others issue more specific tags. Such difference in granularity could be related to the level of user interest, or depth of expertise in the subject (e.g., a general tag as *party* in contrast to a specific tag as *banquet*).

Thus, if we want to use folksonomies for the creation of ontologies we have to tackle the lack of semantics so that tags and relations among them can be used to derive concepts, data and object properties, and instances in an ontology. For ambiguous tags, the proper intended meaning must be identified so that they can be used unambiguously as ontology components. Similarly, different synonyms and morphological variations of a same entity have to be related to a unique concept or to several concepts which are stated as equals in the ontology. Finally, tags representing different level of specificity of the same concept have to be identified as different concepts which are related by a hierarchical relation such as *subClassOf* or thesauri-oriented relations such as *broaderTerm* and *narrowerTerm*.

In the rest of this section we describe the most relevant approaches described in the literature whose main objective is either to extract ontologies from tags in folksonomies or to associate tags to external semantic entities in order to make explicit the meaning of those tags. We identify three groups of approaches according to if they are based on 1) **statistical techniques**, 2) **ontologies**, or 3) on a **hybrid approach** mixing statistical techniques and ontologies. In general, the goal of statistical approaches is to organize tags in a hierarchy or to group related tags in the hope that such grouping will indirectly expose a meaning for their tags. Hence these approaches do not formally define the meaning of tags or their relations. Statistical approaches relies on a notion of **tag relatedness**, and thus we have include a sub-section where we review the different surveys about measures of relatedness between tags. Ontology-based approaches aim at stating the meaning of the tags and their relations by means of associating semantic entities to tags. Hybrid approaches objective can be either 1) to group tags using semantic information, or 2) to associate semantic entities to tags using as context groups of tags.

2.3.2 A unified process for the association of semantics to tags

In this section we propose a unified process that can be used to understand, evaluate and categorize the different approaches for the association of semantics to tags. This process consists of a set of common activities identified in most of the analyzed proposals. The objective is to provide a uniform way to describe the different proposals and therefore to facilitate their assessment.

We designed this unified process following a bottom-up approach. First, we analyzed each approach individually, identifying the activities carried out and main objectives. Then, we highlighted the commonalities between the activities among different approaches and the result was a set of activities that most of the analyzed approaches carry out with different degrees of detail. The activities are presented in a logical sequence, which does not necessarily hold for all the approach. The general process is depicted in Figure 2.3.

Most of the approaches start with defining their data sources, and some of them explicitly describe how they gather information from these data sources. For instance,

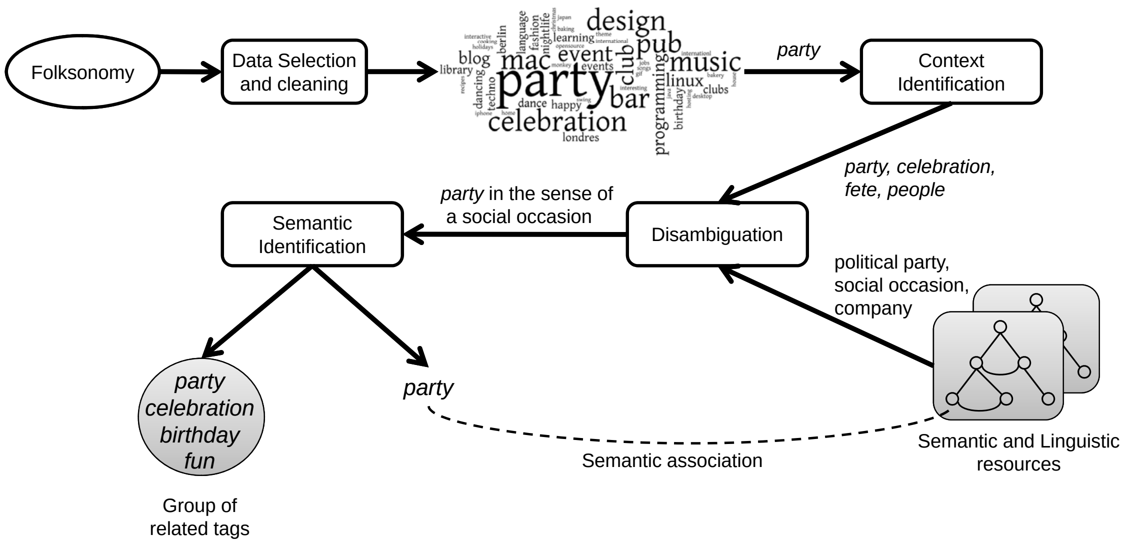

Folksonomy
Data Selection and cleaning
party
Context Identification
party, celebration, fete, people
Disambiguation
party in the sense of a social occasion
Semantic Identification
political party, social occasion, company
party celebration birthday fun
Group of related tags
party
Semantic association
Semantic and Linguistic resources

Figure 2.3: Process for associating semantics to tags (García-Silva et al., 2012)

some research teams designed and developed specialized programs to crawl folksonomies when APIs are not available, or available but with limited coverage and capabilities. For this thesis, the details of how to get the data are not so important. What is relevant are the filters they implement to select and clean the data if they exist. Thus, the first activity identified in our unified process is called **data selection and cleaning**. This activity may include filters that take into account tag use frequency, lexical characteristics like tag length or allowed characters, morphological characteristics, or even the idiom of the tags. With regard to tag frequency, this can be measured based on the number of times the tag has been used to annotate a resource, or the number of times the tag was used by different users.

Once we have the data set we want to work with, the next activity is **context identification** of the tagging activity. In linguistics analysis the context includes features such as part-of-speech labels, collocation information, and surrounding words and sentences (Navigli, 2009). In this thesis context is the set of concepts we take into account to figure out the tag meaning. This will help to identify groups of related tags or to associate a semantic concept to formally define the tag meaning. This notion of context can be applied to the tags of a folksonomy in two ways, the first one uses tags that occur together when they are used to annotate the same resource or group of resources (Mika, 2007), the second one uses tags that are used together by the same user or group of users (Hamasaki et al., 2007). In addition, context can include also tag or resource metadata information such as location coordinates and timestamps (Kennedy et al., 2007).

As mentioned earlier, one of the main problems with folksonomies is that some tags have more than one meaning. This ambiguity yields inaccurate and irrelevant results when these tags are used to search and retrieve information. Therefore, a disambiguation activity (Navigli, 2009) is an important step in the semantic association process. The **disambiguation** activity can be carried out using external semantic resources, such as WordNet, and all the tag context information. Some tag disambiguation approaches, such as those presented in Specia and Motta (2007); Yeung et al. (2007), use clustering techniques in order to group tags according to the resources they annotate or to the users who authored the tags. In either case, according to these methods, if a tag is used to annotate different resource groups or if a tag is used by different user groups, then the tag is considered to have more than one meaning. In general, this type of analysis is more focused on identifying the existence of ambiguity, rather than on identifying the true meaning of a tag.

Finally, the last activity is **semantic identification** in which tag semantics is made formal and explicit. This activity consists of creating mappings between tags and semantic entities, or identifying relations between tags or semantic entities. The matching process between tags and semantic entities like classes or instances is carried out using predefined ontologies or ontologies retrieved at runtime by means of semantic Web search engines. This matching process may result in several semantic entities for a tag (Angeletou et al., 2008), hence aggregation of these entities is required in order to identify which of them refer to the same topic and which does not. In the case that the semantic entities refer to more than one topic, a disambiguation task might be carried out. Furthermore, in this activity could use clustering techniques to identify groups of synonyms, or social network measures, such as clustering coefficient and local centrality (Wasserman et al., 1994), to identify hierarchies of narrower terms and broader terms similar to the relations found in a thesaurus (Mika, 2007).

In addition to the definition of the unified process to compare the different approaches analyzed in the state of the art of this thesis, we will also categorize existing approaches according to the main technique they use. One method for distinguishing between these approaches is proposed by Angeletou et al. (2008), which is based on whether they use statistical clustering techniques to implicitly describe their meaning, or ontology-based techniques to align tags with existing semantic resources. Besides these two categories, we introduce a hybrid category for those approaches mixing clustering and ontology-based techniques. This categorization is useful since most of the proposals based on statistical techniques do not state explicitly the meaning of the tags or the relationships between them, while the ontology-based proposals usually do. On the other hand, hybrid approaches exploit the benefits of statistical and ontology-based techniques to associate semantics to tags or to find groups of related tags.

2.3.3 An illustrative example

In this section we present a simple folksonomy that we will use to illustrate how each of the approaches presented in the following section works. Let us assume that we have the folksonomy shown in Figure 2.4, which consists of four users. User A has no explicit relation with any other user, while user B is explicitly related to users C and D. These relationships are symmetric, as is usually the case in most social networking sites. Although not all tagging systems allow for social relations to be established among users, we will include those relations in the example folksonomy due to the fact that nowadays more and more tagging systems, such as Delicious and Flickr, are supporting social relations.

There are five tags in the example folksonomy: *Coffee*, *Java*, *Language*, *Program and Code*, which are assumed to have been used by our users to tag three resources R1, R2, and R3.

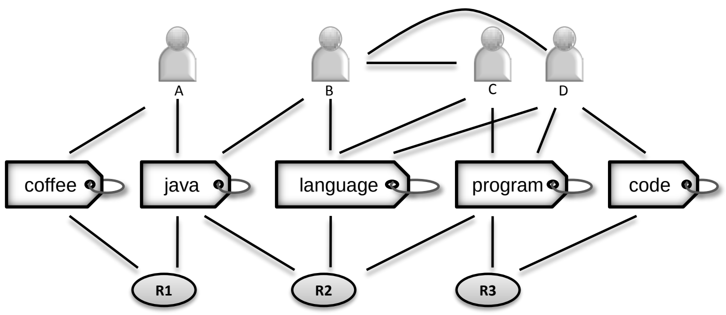


Figure 2.4: Graphical representation of the folksonomy example

The tagging carried out by these users is presented in table 2.2. User A tagged R1 with *Coffee* and *Java*. User B tagged R1 with *Java*, and R2 with *Java* and *Language*. User C tagged R2 with *Language* and *Program*. Finally user D tagged R2 with *Language* and *Program*, and also tagged R3 with *Program* and *Code*.

Table 2.2: Tagging details of the Folksonomy example

Resources	R1		R2			R3	
Users/Tags	*Coffee*	*Java*	*Java*	*Language*	*Program*	*Program*	*Code*
A	X	X					
B		X	X	X			
C				X	X		
D				X	X	X	X

In the following sections we will described the results that can be obtained for this simple example from each of the approaches presented next, so that the similarities and differences between them can be better understood.

2.3.4 Approaches based on tag relatedness measures

Most of the early research about extracting semantic information from folksonomies was based on the definition of ad-hoc tag relatedness measures. For instance Mika (2007) defines that two tags are related if they have been used two annotate a similar set of resources or if they have been used by a similar set of users. These measures were used in the elicitation of ontologies and validated accordingly. Nevertheless other authors (*e.g.*, Cattuto et al. (2008) and Markines et al. (2009)) soon recognized that these ad-hoc measures were just some of the many ways of measuring relatedness and therefore they stated that there was a lack of surveys regarding tag relatedness measures and the semantics that these relations convey. Thus they propose different tag relatedness measures and evaluate them systematically in terms of the semantics that can be elicited with these measures. In this context, the relatedness measures were defined according to two dimensions. The first dimension corresponds to how tags are represented (*e.g.*, as plain labels or using vector space model), and the second one is how tags are compared so that they can be classified into related or not related (*e.g.* using jackard's coefficient (Jaccard, 1901)). In this section we describe research works presenting surveys about tag relatedness measures. These works are the foundation over which statistical-based approaches to discover the emergent semantics from folksonomies are designed.

2.3.4.1 Cattuto et al.'s approach

Cattuto et al. (2008) define five relatedness measures between two tags: *co-occurrence, tag context, resource context, user context, and FolkRank.* **Co-occurrence** between two tags measures the number of posts containing them. Thus two tag are considered related is the post number in which they co-occur is maximal. **Tag context** measure is calculated in the vector space $\Re^T$, where T is the set of tags. For a tag t_i each vector entry v is the number of posts containing both t_i and the corresponding entry tag t in the vector. **Resource context** measure is calculated in the vector space $\Re^R$ where R is the set of annotated resources. For a tag t_i each vector entry v is the number of times that the corresponding resource is tagged with t_i. **User context** measure is calculated in the vector space $\Re^U$ where U is the set users. For a tag t_i each vector entry v is the number of times that the corresponding user use the tag t_i. In case of tag context, resource context, user context relatedness between two tags is defined by means of the cosine measure. Finally, **FolkRank**, which is based on the PageRank (Brin and Page, 1998) algorithm, produces a ranked list of relevant tags for a given tag.

Authors of this work conclude that tag context and resource context measures identify more synonyms and spelling variants. They also conclude that FolRank measure yields more general tags.

2.3.4.2 Markines et al.'s approach

Markines et al. (2009) define different types of tag representation: *projection, distributional, macro-aggregation and collaborative.* In **projection** tagging triples $<u,t,r>$ are transformed into tuples $<t,r>$ thus getting rid of the user dimension. In **distributional** tagging triples are turn into $<t,r,n>$, where n is the number of users that have annotated r with t. In **macro aggregation** per each user u the tagging triples are turn into $<t,r,m>$, where m is 1 if the user u has annotated r with t and 0 otherwise. User triples are used to calculate a user-based similarity σ between two tags. Thus the final similarity is calculated by summing the individual user similarities. This guarantee that every user contributes in the same way to the final aggregation. Finally **collaborative** is similar to macro-aggregation though in this case user resources are required to have at least one annotation in common. This is achieved by adding an artificial tag to all the user resources. In this case the user-based similarity σ yields positive values for all pair of user tags.

For each type of tag representation authors define the following similarity measures: *Matching, Overlap, Jackard, Dice, Cosine and Mutual information.* In the **projection** case matching similarity is defined as the number of resources annotated with both tags. **Overlap similarity** is the fraction of resources annotated with both tags with respect to the minimum number of resources annotated with each tag. **Jaccard coefficient** is the number of resources annotated with both tags divided by the sum of the number of resources annotated with each tag. **Dice coefficients** is similar to the Jaccard coefficient but has some different properties. **Cosine similarity** is measured between the two vectors in $\Re^R$ representing the two tags. Finally, the **mutual information** is calculated according to its definition used in information theory (Shannon, 1948). Authors also provide the definition of each of these measures for the distributional, macro-aggregation and collaborative representations.

Authors of this work conclude that mutual information is the measure that best extracts semantics similarity information. They also stated that macro-aggregation produce less semantic relations than the distributional representation. Though not as good as the aforementioned, collaborative representation is able to extract an important amount of semantic relationships.

2.3.4.3 Körner et al.'s approach

Körner et al. (2010) evaluates how a tag similarity measure known as *tag context* behaves when it is measured on subsets of folksonomies defined by users that are identified as categorizers and describers. **Tag context** similarity is defined in the same way than in (Cattuto et al., 2008). That is, each tag is represented by a vector in $\Re^T$, and similarity between two tags is calculated by measuring the cosine of the corresponding vectors. On the other hand, to differentiate between categorizers and describers they propose a set of measures based, for instance, on the number of tags of each user, or on the number of user tags divided by the number of resources annotated by the user, or on the average number of tags per posts. Users are sorted according to the values of these measures so that describers are in one extreme of the list and categorizers in the other. Then different subsets of the folksonomy are created by selecting the annotations of subsets of describers and of categorizers. These describers and categorizers subsets are defined according to the percentage of folksonomy users desired in each category. Finally the tag context measure is calculated per pair of tags in each of the sub-folksonomies.

Authors found that describers, with the exception of some spammers, generates subsets of folksonomies for which is possible to identify more semantically related tags according to the definition of the tag context similarity measure.

2.3.4.4 Benz et al.'s approach

In (Benz et al., 2011) authors focus on a especial case of relatedness which is defined by hierarchical relations between two concepts. Examples of these relations are subsumption relations, and broader or narrower term. To do so, they define different tag generality measures (or abstracness measures): *frequency-based, entropy-based, centrality,* and *statistical sumbsuption.* **Frequency-based** is a ranking function where tags are ordered according to the number of posts where they have been used. **Entropy-based** assumes that more abstract terms have a higher entropy since they are probably used at a relatively constant level to annotate a broad range of resources. **Centrality** assumes that more central tags in the folksonomy graph are more general tags. In this work centrality is measured in three ways: *degree*, *betweeness*, and *closeness*. In general for a given node n in a graph, **degree** is the number of direct nodes connected to n, **betweeness** is the fraction of shortest paths between each pair of nodes in the graph passing trough n, and **closeness** is defined as the inverse of the *farness* which in turn is defined as the sum of its distance (shortest path) to all other nodes. These centrality measures are applied on two different graphs. The first graph connects two tags if they have been concurrently used by an user in a post. The second graph connects two tags if they are related according to the *resource-context* measured defined in (Cattuto et al., 2008). Finally **statistical subsumption** assumes that a tag t subsumes another tag t' if $p(t|t') > \epsilon$ and $p(t'|t) < \epsilon$.

Authors conclude that measures based on frequency and entropy are correlated with results extracted from gold-standard taxonomies. They also note that centrality measures on the graph created from tag co-occurrence and the subsumption probabilistic model yield an important amount of results correlated with the gold standard taxonomy. Finally, they mention that even the measure based on the tag frequency produces a considerable correlation to the gold standard taxonomies.

2.3.5 Statistical-based approaches

Several approaches exist, whose goal is to identify the semantics of tags, that propose to cluster tags according to some relations among them (Begelman et al., 2006; Hamasaki et al., 2007; Jäschke et al., 2008; Kennedy et al., 2007; Mika, 2007). Some other works attempt to organize the tags in a hierarchy (Benz et al., 2010; Heymann and Garcia-Molina, 2006). In both cases thees approaches relies on tag relatedness measures (see section 2.3.4). Approaches discovering tag clusters use tag relatedness measures so that related or similar tags are grouped together. On the other hand approaches generating hierarchies distinguish between general and specific tags to create the hierarchies. In this section we present research works that exploit tag relatedness measures to find groups of tags as those depicted in Figure 2.5 or to generate tag hierarchies.

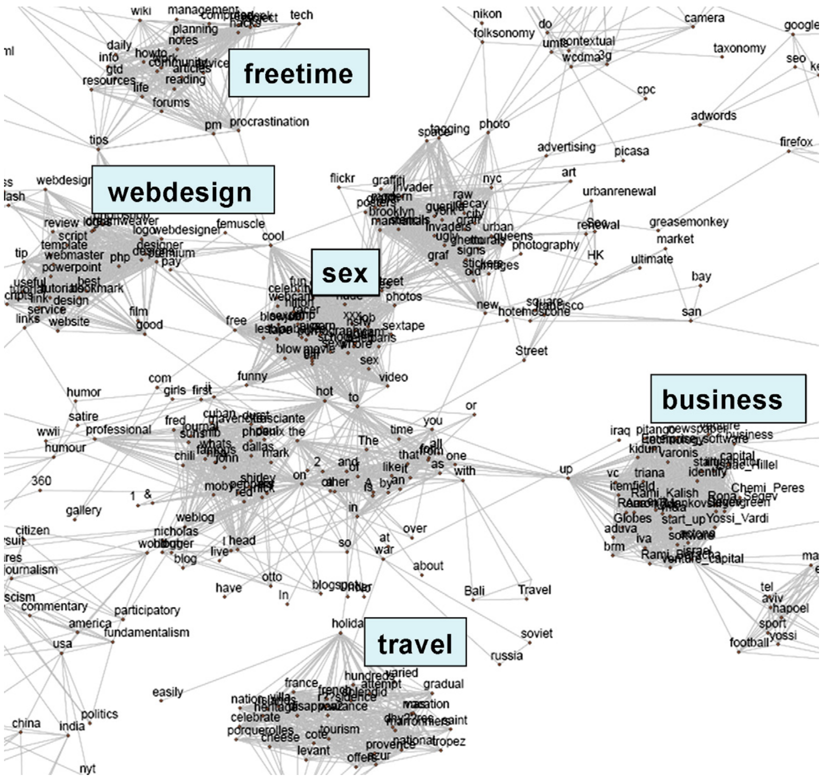


Figure 2.5: Tag clustering based on tag co-occurrence when annotating resources (Mika, 2007)

2.3.5.1 Mika's approach

Mika describes an approach to generate two lightweight ontologies from folksonomies using statistical techniques (Mika, 2007): an ontology of concepts based on the overlapping set of user communities (O_{ac}), and an ontology of concepts based on the overlapping set of resources (O_{ci}). The approach is tested with two data sources; a set of users, terms and Web pages from the Semantic Web research community, and a folksonomy extracted from the Delicious website. We use the latter case to describe this approach following our uniform process detailed in Section 2.3.2.

Data Selection and Cleaning: The data selection and cleaning activity of this approach is limited to filtering out from the folksonomy those tags with less than ten items classified under them and those persons who have used less than five tags. The Delicious dataset used to test the approach consists of 51,852 anotations of 30,790 resources by 10,198 users using 29,476 distinct tags.

Context Identification: The context identification activity is carried out in a distinct way for each of the resulting ontologies. In the case of the ontology O_{ci}, context is defined as the tags that co-occur with a particular tag when they are used to annotate a resource. In the case of the ontology O_{ac}, context is defined as the tags that co-occur with a particular tag when they are used by a user or group of users.

Disambiguation: This approach does not explicitly deal with disambiguation problems.

Semantic Identification: Finally, the semantic identification activity is also carried out differently for each of the ontologies that this approach generates. A graph of concepts is built for O_{ci} where the edges indicate that two concepts (tags) were used together when annotating one or more resources. These links are weighted by the number of resources annotated using both tags. Similarly, a graph of concepts is also built for O_{ac} where the edges indicate that two concepts were used together by one or more users. These links are weighted by the number of people who have used both tags. After this graph generation process, clustering is performed to identify specific words inside each cluster and general words bridging different clusters. Set theory principles are then applied to define relations between concepts as broader and narrower terms.

The output is a hierarchy of concepts O_{ac} based on subcommunity relationships and a classification hierarchy O_{ci} based on resource overlap.

Using our folksonomy example from Section 2.3.3, the ontologies obtained with this approach are presented in Figure 2.6. In O_{ac} the *Language* and *Code* tags are related because user D used them to tag R2 and R3 respectively, while in O_{ci} these tags are not related because they were not used to tag the same resource. On the other hand in O_{ci} tags *Java* and *Program* are related because these tags were used to tag the same resource R2, while in O_{ac} these tags are not related because they were not used together by any user. Then, if we want to discover in more detail the relations between these tags, we have to analyze the tag set using set operations. For instance, let us assume that we are interested in discovering the relation between *Java* and *Language* in the case of O_{ci}. If all the resources annotated with the *Java* tag are included in the set of resources annotated with the *Language* tag, and this last set is large enough, then according to this approach *Language* is broader than *Java*, meaning that all resources classified under *Java* are also classified under *Language*.

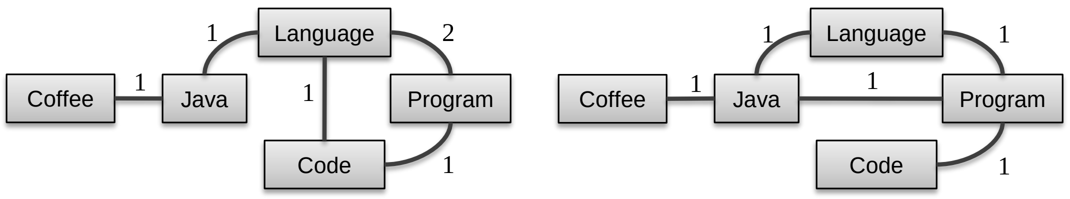


Figure 2.6: Tags relations found by Mika's approach; (a) Left side: O_{ac} tag are related if used by same user, and (b) Right side: O_{ci} tag are related if used to tag the same resource.

This approach allows reflecting the ongoing behavior of folksonomies for a set of user communities. However, there are some limitations to this approach. Tag ambiguity is one of the main problems present in folksonomies, as described in the introduction, and it is not clear in this approach how ambiguous tags can affect or be reflected in the generated ontologies. Furthermore, the identified relations mostly represent co-ocurrence, rather than any ontological relation such as *subclass of* or *part of*. Finally, there is no explicit catering for misspelled tags, or for tags in morphological variations.

2.3.5.2 Hamasaki et al.'s approach

Hamasaki and colleagues extended Mika's work with the notion of user neighborhood, which can be described as the direct contacts of a user (Hamasaki et al., 2007). Particularly, the O$_{ac}$ ontology is modified by taking into account tagging information of the user neighbors in the folksonomy. The data source used by Hamasaki *et al.* to evaluate this approach is a folksonomy adapted from a community support system for academic conferences, where users can bookmark documents of interest.

Data Selection and Cleaning: No general rules for data selection and filtering are described for this approach. The dataset used to test the approach comprises 314 tags, 297 resources, and 75 users who have bookmarked at least one resource, and a total of 323 users in the social network.

Context Identification: Context is shaped from the user tags along with the tags used by his neighbors. Tagging information of neighbors could help to overcome any lack of tagging information for a particular user.

Disambiguation: Unlike Mika's approach, this approach proposes an algorithm for disambiguation. The algorithm is based on the idea that if a tag is used to annotate different resources by different groups of users (neighbors), the tag may have different meanings. Otherwise, the tag has only one (or very similar) meaning. The proposed algorithm treats each user tag as a pre-concept, and then these pre-concepts are merged if they have the same labels and share the same users/resources or neighboring users.

Semantic Identification: There is no description in this approach of what concerns the semantic identification activity. As pointed out above, each pre-concept previously identified and possibly merged with others is converted into a concept. However, the relations between those concepts are not explicitly defined.

Figure 2.7 shows the O'$_{ac}$ ontology that is obtained when applying this approach to our sample folksonomy. The O$_{ci}$ ontology remains the same as the one described in Mika's approach. If we compare this O'$_{ac}$ ontology with the O$_{ac}$ ontology described in Mika's approach, we can see that two new relations appear. First, the *Java* and *Program* tags are related even though they are not used by the same user. In this case, user B is related to user C, and following this approach, the tags from user C are considered to be indirect tags for user B. Therefore, user B uses *Java* and *Program* tags, being

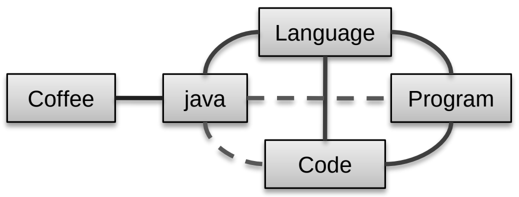


Figure 2.7: O_{ac} tags are related if used by the user or its social network.

the *Program* tag taken from user C. Secondly, the relation between *Java* and *Code* is established from the relation between users B and D.

This approach presents some advances over Mika's approach. However, the data source used to evaluate the approach, as described in Hamasaki et al. (2007), has some characteristics that are not present in real folksonomies. In this data source owners of resources assign tags to them. However, if a user, other than the resource owner, bookmarks these resources, then the tags assigned previously by the owner are considered to have also been assigned by this user. Thus, all users who bookmark a document share, in a mandatory way, the same vocabulary. In contrast, users in major existing folksonomies assign tags freely, and the convergence of a vocabulary for a resource occurs when several users tag the same resource using a tag recommendation strategy that includes the most popular tags used by other users to tag the same resource. We want to note that besides recommendations based on tag popularity, other tag recommendation strategies have been developed based on collaborative filtering (Jäschke et al., 2008), and association rules (Schmitz et al., 2006), among others.

The proposed disambiguation strategy is strongly influenced by the group of users selected and how they tag. However, as explained above, the data source used by the authors of this approach is biased because all users who bookmark a document are assumed to share the same tags, and hence it is unclear whether this strategy would be adequate for open environments. Finally, the approach does not propose any method to semantically define the meaning of tags and their relationships.

2.3.5.3 Jäschke et al.'s approach

Jäschke and colleagues proposed a statistical approach for discovering subsets of users who implicitly agree on common tags for a set of resources in folksonomies (Jäschke et al., 2008). The approach is tested with three data sources; Delicious and BibSonomy[1] folksonomies, and a non folksonomy application called *IT baseline security manual.*

Data Selection and Cleaning: The data selected for the experiments is a snapshot of the chosen folksonomies. For instance, in the case of Delicious, authors used as their dataset all tagging information entered into the system before June 16, 2004. This dataset contained over 3.3K users, around 30.5K unique tags, over 220K resources, with close to 617K links. With respect to Bibsonomy, the snapshot included all data up to November 23, 2006, excluding any automatic insertions (such as DBLP publications) as well as any automated default tags (such as *imported*). The Bibsonomy testbed contained almost 45K tag assignments, 262 users, and over 11K resources (publications) tagged with close to 6K distinct tags. As for the IT security manual dataset, this was set up by a closed group of experts and not by an open folksonomy, and used by the authors as an ontology for analyzing their approach, rather than for discovering any tag semantics.

Context Identification: Context identification in this approach consisted of mining all frequent tri-concepts over the selected information in order to obtain a set of triples, where each triple contains a set of users, a set of tags and a set of resources. Each user in the set of users has tagged each resource in the set of resources with all the tags in the set of tags.

Disambiguation: Disambiguation of the tags used in these triples is not addressed in this approach.

Semantic Identification: The semantic identification of tags is carried out by selecting from the triples found in the previous activities those tag sets which we are interested in. Then for each tag set a concept lattice is created. A concept lattice is a hierarchical conceptual clustering of tags. The formal context of the concept lattice is composed of resources tagged with at least one of the tags in the tag set by a particular

[1]BibSonomy is a social bookmarking and publication sharing system. `http://www.bibsonomy.org/`

number of users, and of tags which were used by the majority of users in a resource. The graphic representation of this concept lattice could then be used by ontology engineers to manually build a concept hierarchy that corresponds to the original folksonomy.

With respect to our running example folksonomy, let us extend the tagging of users C and D with the *Java* tag to annotate R2. Table 2.3 shows the shared conceptualizations that can be found in this folksonomy. Users B,C and D agree on the use of *Java* and *Language* as tags for R2. Users C and D agree on the use of *Java*, *Language* and *Program* as tags for R2. Users A and B agree on the use of *Java* to tag R1. According to the authors the sets of users, tags and resources of each shared conceptualization have the property that none of them can be extended without shrinking one of the other two sets. That is, if we want to expand the tag set in which users B, C and D agree to annotate R2 with the tag *Program*, users B will be eliminated from the user set.

Table 2.3: Groups of users, tags and resources

	Shared Conceptualizations		
Users	B,C,D	C,D	A,B
Tags	*Java*, *Language*	*Java*, *Language*, *Program*	*Java*
Resources	R2	R2	R1

As an example, from these triples we can create a concept lattice to analyze the tags *Java*, *Language*, *Program*. Let us suppose that the tag *Java* has been used only together with the *Language* tag, while the *Language* tag has been used independently in other cases. Thus, in the graphical representation of the concept lattice the *Language* tag will be above the *Java* tag, meaning that all the resources tagged with *Java* are also tagged with *Language*, and that *Language* has been used in a more general sense to annotate another group of resources. This hierarchical relation can be then analyzed by an ontology engineer to find the appropriate semantic relation between *Java* and *Language* tags.

Triples and the corresponding concept lattices that are generated by applying this approach could provide useful information for ontology construction. However, on the one hand, in this approach two different triples can be generated including the same tag set, but for different sets of users and resources. This will be the case if some people use the tag set to annotate a resource set, while some other people use the same tag set to annotate another resource set. Nevertheless the resource sets could be related, for instance if they are about the same topic, and thus the shared conceptualization could be extended to cover all users and resources. On the other hand, the output of the process is a hierarchical representation of tags, but the relationships between tags in different hierarchical levels are not defined semantically, and this task is left to an ontology engineer. Finally, in this approach there is no strategy to deal with ambiguous tags.

2.3.5.4 Limpens et al.'s approach

Limpens and colleagues proposed a statistical approach for enriching semantically folksonomies (Limpens et al., 2010). They enrichment consist of suggesting spelling variations of a tag, other related tags, and more general and more specific tags. The approach is tested with data extracted from a folksonomy generated within a company.

Data Selection and Cleaning: This activity is not considered in this approach. Nevertheless, authors mention that they hand picked 88 tags from the input folksonomy to test their approach.

Context Identification: Authors defined two different context. First context definition is called tag-context (Cattuto et al., 2008) and it is based on co-occurring tags. They represent each tag as a vector of tags and each position in the vector has as value the frequency of co-occurrence of the tag being represented and the tag corresponding to that position in the vector (see section 2.3.4.1). The second context for a tag is defined as the users who have used it. First context is used to identify related tags and second one is used to identify subsumption relations as we will show in the Semantic Identification activity.

Disambiguation: This approach does not explicitly deal with ambiguous tags.

Semantic Identification: Authors propose to use string similarity metrics over tags and the folksonomy structure to detect spelling variations, related tags and hyponym relations. They evaluated 30 different string similarity metrics, and based on the results of this study authors propose the following heuristic algorithm. They algorithm starts by collecting related tags using the *Monge-Elkan_Soundex* metric. Then they use the

JaroWinkler similarity to see if the related tags are spelling variants. If it is not the case, authors propose to use the *MongeElkan_ QGram* metrict to identify hyponym relations. According to the reported results this algorithm is good when identifying spelling variations and not that good when detecting related tags and hyponyms. Thus, authors propose to use other existing approaches exploiting the structure of the folksonomy to identify related tags and hyponyms. To identify related tags they use the tag-context similarity measure (see section 2.3.4.1). To identify hyponyms they use Mika (2007) approach, which is based on measuring the overlap of users of each tag. The intuition is that a tag t_i is more general than a tag t_j if the set of users U_j using t_j is included in the set of users U_i using t_i. This can be measured using: $overlap(U_i, U_j) = \frac{|U_i \cap U_j|}{|U_i|}$.

If we apply the string similarity metrics proposed in this approach to our running example we are not going to find related tags since the set of tags are not similar according to their string representations. If we apply the tag-context similarity metric and set a minimum threshold of 0.6 we found that the tags *Java* and *Program*, and *Language* and *Code* are related. If we apply the heuristic proposed by Mika (2007) with a minimum threshold of 0.6 we found that *Language* is more general than *Program* since users B, C, and D use *Language* while users C, and D use *Program*, thus the overlap of the user sets is 0.67.

2.3.5.5 Other statistical-based approaches

Other clustering algorithms have been proposed to identify groups of related tags. Though the main goal is to group tags according to certain characteristics, they results are rather limited with respect to the semantics of the groups, and therefore we describe them briefly. We present three approaches: 1) Begelman et al. (2006) cluster tags according to tag co-occurrence when annotating resources, 2) Kennedy et al. (2007) cluster tags based on time and location metadata, and 3) Heymann and Garcia-Molina (2006) generates a hierarchy of tags where the branches represent topical clusters of tags. We also presents the modifications proposed by Benz et al. (2010) to this last approach.

Begelman et al. (2006)'s approach proposes to create a graph where the vertices are tags, and edges between two tags exist if they co-occur in the annotations of one or more resources. These edges are weighted by their co-ocurrence frequency. A technique called Spectral bisection is used to split this graph in to clusters, and a modularity function is used to compare the quality of the new clusters against the previous one to evaluate whether the new clusters are accepted or not. This technique is executed recursively on the new clusters.

On the other hand, Kennedy et al. (2007)'s approach presents a clustering-based technique to identify tags related to locations and events. The approach relies on the latitude and longitude of the geotagged resources as well as on the timestamp. Each tag has an associated spatial and temporal distribution. The spatial distribution is a list of the geographical coordinates of the pictures annotated with that tag, and the temporal distribution is a list of the timestamps where the picture was taken or when it was uploaded to the system. To identify location and event tags, a clustering algorithm was applied to the spatial distribution to find groups of tags sharing spacial patterns, and on the temporal distribution to find tags sharing temporal patterns.

Heymann and Garcia-Molina (2006) presents a greedy algorithm to create tag hierarchies. In this approach tags are represented using vectors, where each position represents the times the tag was used to annotate a resource. Those vectors are compared using as similarity measure the cosine of the angle they form. With this information a similarity graph is built, where the vertices are tags, and edges are weighted by the similarity measure calculated previously. Next, for each tag the network centrality is calculated in the similarity graph. Finally, the tags are ordered in a list according to this centrality value. This list is processed, starting with the most central tag, to create the tag hierarchy. Heymann and Garcia-Molina (2006) work has been extended in (Benz et al., 2010). Benz et al. (2010) propose to first preprocess the folksonomy so that synonyms are grouped using a tag-context similarity measure (see section 2.3.4.1). This preprocessed folksonomy is used as the input of the algorithm to generate the hierarchy. Authors also propose to identify ambiguous tags by applying a clustering algorithm to each tag context. Each cluster found is a possible meaning. The context for each tag is created with the most frequently co-occurring tags. Thus, in the main algorithm when they are considering a tag to be included in the hierarchy they identify if the tag is ambiguous and attempt to disambiguate it before including it in the hierarchy.

In short Begelman et al. (2006)'s output are groups of related tags, Kennedy et al. (2007)'s output are groups of tags identified as locations or events, and Heymann and Garcia-Molina (2006)'s output is a hierarchy of tags. A drawback shared by Begelman et al. (2006) and Heymann and Garcia-Molina (2006) works is that they do not identify the semantics of the clusters, the tags, nor the relations between tags. Kennedy et al. (2007)'s approach is a step forward to the identification of tag semantics using clustering techniques based on tag metadata. Nevertheless, this approach is limited to two semantic classes (*i.e.*, locations and events). Furthermore, relations between the tags belonging to the same or different groups is not established.

2.3.6 Ontology based approaches

The approaches presented earlier mainly focus on applying statistical techniques to group tags or to show relatedness between them. In addition to these approaches, there exists a number of approaches aiming at associating semantic entities to tags as a way to formally define their meaning (Angeletou et al., 2008; Cantador et al., 2008; García-Silva et al., 2009; Maala et al., 2008; Passant, 2007; Tesconi et al., 2008). In this section, we review these works.

2.3.6.1 Angeletou et al.' approach

Angeletou and colleagues proposed an automatic approach to enrich folksonomy tags with formal semantics by associating them with relevant concepts defined in online ontologies (Angeletou et al., 2008). The data source used to test the approach is a Flickr data set.

Data Selection and Cleaning: The initial Flickr dataset comprised 250 resources, and 2819 tags. As in the previous approach, during the data selection and cleaning activity, some tags are filtered out, including numbers, special characters and non English tags. The main reason to eliminate these tags is that this approach relies on WordNet, a lexical database for the English language, and usually its entries do not include numbers or special characters.

Context Identification: The filtered tags then go through context identification. For each tag, all of its possible lexical representations, such as singular, plurals, or various delimited types of compound tags, are generated. The context is defined as the whole filtered tag set along with their lexical representations.

Disambiguation: Tags and their contexts are taken as input to the disambiguation activity. In this activity, if a tag has more than one sense in WordNet, then the hierarchy of its senses as extracted from WordNet is used to calculate the similarity with the senses of all tags in the tag set and thus disambiguating them. The similarity between senses is calculated using the Wu and Palmer similarity measure (Wu and Palmer, 1994).

Unlike other approaches described so far, in this approach there is a phase of context expansion after disambiguation. For each tag, synonyms and hypernyms are extracted from WordNet using the sense assigned in the disambiguation phase. This information is used later to find the right ontology entity that will be associated with each tag.

Semantic Identification: The semantic identification activity in this approach focuses on relating the expanded set of tags to ontological entities, using the Watson[1] semantic search engine. For each tag, several ontological entities may be retrieved, which are then integrated in order to group similar ontological entities. The similarity measure used for this integration process compares the entity labels and the semantic neighborhood information including superclasses and subclasses. Finally, tags are associated with one or more ontological entities, comparing the ontological parents of the merged entities with the tag hypernyms.

The whole process proposed is depicted in Figure 2.8. Table 2.4 presents the result of applying this approach to some of the tags in our sample folksonomy. In this case, the most probable sense of the *Java* tag is Java as a programming language because most of the tags are related with this sense. Thus, the *Java* tag is associated with an instance of the class *Java_(programming_language).* The *Code* tag is associated with the *ComputerCode* class which is a *subclass of* the *DigitalResource* class. The *Program* tag is associated with the class *Program* which is a *subclass of* the *ProgrammingSoftware* Class.

Angeletou and colleagues evaluated their approach, achieving a high precision rate of 93%, with an initial low recall rate of 24.5%, which then was raised to 49% by estimating how many of the tags actually could be related with ontological entities using the information provided by Watson. This evaluation was carried out using 226 photos whose tags were enriched semantically using this approach, and the associations between tags and ontological entities were manually checked.

[1] http://watson.kmi.open.ac.uk/

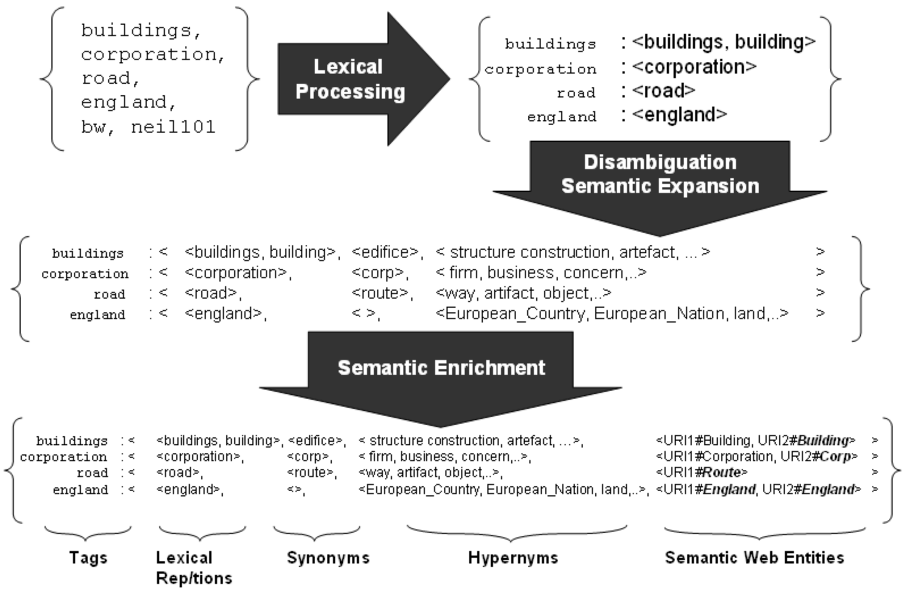

buildings, corporation, road, england, bw, neil101
Lexical Processing
buildings : <buildings, building>
corporation : <corporation>
road : <road>
england : <england>
Disambiguation Semantic Expansion
buildings : < <buildings, building>, <edifice>, < structure construction, artefact, ... > >
corporation : < <corporation>, <corp>, < firm, business, concern,..> >
road : < <road>, <route>, <way, artifact, object,..> >
england : < <england>, < >, <European_Country, European_Nation, land,..> >
Semantic Enrichment
buildings : < <buildings, building>, <edifice>, < structure construction, artefact, ...>, <URI1#Building, URI2#Building> >
corporation : < <corporation>, <corp>, < firm, business, concern,..>, <URI1#Corporation, URI2#Corp> >
road : < <road>, <route>, <way, artifact, object,..>, <URI1#Route> >
england : < <england>, <>, <European_Country, European_Nation, land,..>, <URI1#England, URI2#England> >
Tags
Lexical Rep/tions
Synonyms
Hypernyms
Semantic Web Entities

Figure 2.8: Semantic enrichment process (Angeletou et al., 2008)

Table 2.4: Tags and related semantic entities

Tags	Semantic entity
Java	http://dbpedia.org/resource/Java *typeOf* http://dbpedia.org/resource/Java_(programming_language)
Code	http://www.lt4el.eu/CSnCS#ComputerCode *subClassOf* http://www.loa-cnr.it/ontologies/IOLite.owl#DigitalResource
Program	http://www.lt4el.eu/CSnCS#Program *subClassOf* http://www.lt4el.eu/CSnCS#ProgrammingSoftware

These results are promising, since they show that these types of automatic techniques can achieve good results. However, there are also some limitations. For instance, this approach tries to find one sense for a tag. While this could be valid for a particular tag set, there is the possibility that an ambiguous tag is used in more than one sense. The authors also mention that some tags and their context were not found in the WordNet hierarchy of senses, and thus the disambiguation activity failed. WordNet is limited in terms of its terminological coverage, and hence seriously limits the scope of this approach. Finally, the semantic identification activity tries to match hypernyms of WordNet with ontological parents. WordNet is a lexical database and its hierarchy of hypernyms does not necessarily correspond to the hierarchy of concepts found in ontologies.

2.3.6.2 Cantador et al.'s approach

Cantador and colleagues present an automatic approach to associate folksonomy tags with domain ontology concepts using Wikipedia[1] categories as an intermediate shared representation between tags and ontology classes (Cantador et al., 2008). The data source used to test the approach is a tag set extracted from Delicious and Flickr.

Data Selection and Cleaning: The initial tag set contains 28,550 tags, gathered from Delicious and Flickr. The data selection and cleaning activity, depicted in Figure 2.9, focuses on filtering out tags which are too small (one letter tags) or too large (tags with more than 25 characters). Moreover, special characters are converted to their base form (e.g., ü is converted to u), and tags with a low frequency or that are common stop-words are removed. Then, each tag is searched in WordNet. If a tag does not exist in WordNet, then the Google *did you mean* mechanism is used to correct any possible tag misspellings, or to break up any compound tags (tags made up of concatenated words). Otherwise, the tags are assumed to be acronyms, abbreviations or proper names. In the latter case, those tags are searched in Wikipedia for an agreed representation. Furthermore, morphologically similar tags are grouped in a single tag using a singularisation algorithm and a stemming function, and the shortest term in WordNet is used as the representative tag. Finally, tags which are non-ambiguous synonyms are merged. The synonym information of each tag is retrieved from WordNet.

[1] http://en.wikipedia.org/

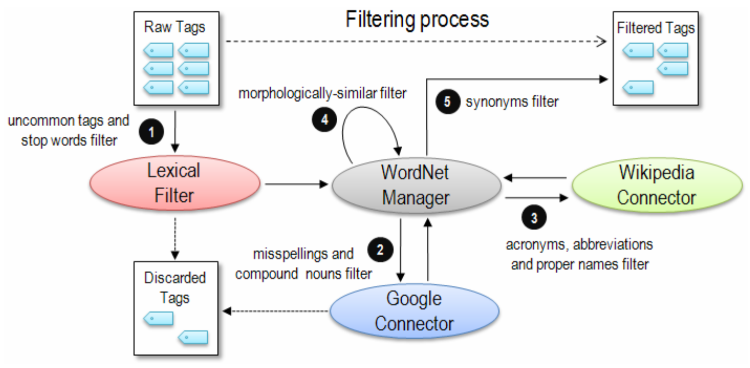

Raw Tags
Filtering process
Filtered Tags
uncommon tags and stop words filter
1
Lexical Filter
morphologically-similar filter
4
5 synonyms filter
WordNet Manager
Wikipedia Connector
3
acronyms, abbreviations and proper names filter
misspellings and compound nouns filter
2
Discarded Tags
Google Connector

Figure 2.9: Tag filtering process (Cantador et al., 2008)

Context Identification: The context identification activity retrieves information from Wikipedia for each tag, including the Wikipedia page URL, and the Wikipedia category list for that page.

Disambiguation: The disambiguation activity is not specified in this approach, although Wikipedia disambiguation pages are pointed out as a possible source of information to disambiguate tags.

Semantic Identification: During semantic identification each tag is given a concept URI using the tags context information, which includes the Wikipedia page name and the Wikipedia category previously associated with each tag. To this end, the terms in the context, that is the terms in the category names, are compared against the domain ontology classes, and the most appropriate ontology classes among the matching ones are selected. Finally, an instance of each of the ontology classes is created. The URI of those instances is the Wikipedia page name, and the categories are assigned as instance labels.

In our sample folksonomy, all the tags are ambiguous according to WordNet and Wikipedia. However, Wikipedia only displays a disambiguation page for the *Program* tag, while for the other tags the most probable page is displayed and in this page there is a link to the disambiguation page. Therefore, the *Program* tag is not processed for this approach because of its ambiguity. The *Coffee* tag is related to the Coffee Wikipedia page which refers to the coffee beverage. This page has the categories: Coffee, Arabic culture, Arabic loanwords and Crops. In this case, the Coffee Wikipedia category will be selected since it matches exactly the *Coffee* tag. Therefore an instance of this class will be created using the Wikipedia page name as its URI (as shown in Table 2.5). However, for the other tags the Wikipedia pages displayed are not in the context of programming languages. For instance, for the *Java* tag the Wikipedia page refers to Java as the island in Indonesia.

Table 2.5: Semantic association between tags and ontology concepts

Tags	**Semantic entity**
Coffee	http://en.wikipedia.org/wiki/Coffee *typeOf* http://mydomain.org/userPreferencesOntology/Coffee
Java	http://en.wikipedia.org/wiki/Java *typeOf* http://mydomain.org/ProgrammingLangOntology/Java
Language	http://en.wikipedia.org/wiki/Language *typeOf* http://mydomain.org/ProgrammingLangOntology/FormalLanguage

According to the authors of this approach, the advantage of using Wikipedia as a shared representation for tags is that Wikipedia is maintained collaboratively by a large user community. Thus, Wikipedia incorporates new terminology faster than linguistic resources like WordNet. However, this approach fails when the Wikipedia page is not directly related with the intended meaning of the tag according to its context, mainly because the approach lacks a disambiguation process.

2.3.6.3 Tesconi et al.'s approach

Tesconi and colleagues' approach is based on mapping tags to Wikipedia pages and then associating those tags with other semantic resources (Tesconi et al., 2008). The approach has been tested using tagging information retrieved from Delicious.

Data Selection and Cleaning: Tagpedia[1] is used as a sense repository to find the set of candidate Wikipedia pages related to a particular tag. Tagpedia associates terms to Wikipedia pages by gathering information from Wikipedia disambiguation and redirection pages. Thus, morphological variations as well as synonyms are implicitly managed.

[1] http://www.tagpedia.org/

The data set consists of the tagging information of nine Delicious users comprising 3,520 tags used to annotate 3,926 resources.

Context Identification: The context of a tag consists of the user tags co-occurring with the tag when annotating any resource tagged by the user, plus the Delicious most popular tags for the set of resources annotated by the user.

Disambiguation: In this approach, the disambiguation activity calculates for each relevant Wikipedia page associated to an ambiguous tag a sense-rank value and selects the one with the highest value. The sense-rank value is calculated by taking into account co-occurrence or popularity frequency of each tag in the context. Co-occurrence is used when the tag in the context co-occurrs with the ambigous tag. On the other hand, popularity is used when the tag in the context was extracted from Delicious popular tags. In addition, the sense rank value includes the number of occurrences of each one of the tags in the context in the analized Wikipedia pages.

Semantic Identification: In the semantic identification activity the selected Wikipedia page is used to find the Wikipedia categories containing that page, and the corresponding DBpedia resource. From DBpedia resources authors extract references to YAGO ontology [1] classes and WordNet Synsets.

The results of applying this approach to our folksonomy example are presented in Table 2.6. These results are then extended with YAGO concepts and WordNet synsets. For each DBpedia resource in table 2.6 we looked for the corresponding YAGO concept. We did not found any YAGO concept related to the dbpedia resources *Coffee*, *Java_ coffee*, and *Programming_ language*. Nevertheless, we found a YAGO relation for the dbpedia resource *Java_ (programming_ language)* stating that it has an *owl#sameAs* relation with the concept *yago:Java_ (programming_ language)*

Authors of this approach produced in their evaluation of the disambiguation process a 89,15% of correct disambiguations of distinct polysemous tags. 11,71% of the tags have not been associated to any Wikipedia page. In addition, they evaluated the coverage of Wikipedia categories, YAGO classes and WordNet synsets, and produced 95%, 58% and 18% accuracy respectively. The accuracy of the disambiguation process seems very

[1] A semantic knowledge base created from Wikipedia information http://www.mpi-inf.mpg.de/yago-naga/yago/

Table 2.6: Semantic association between tags and DBpedia resources

User	Tags	Semantic entity	Resource
A	*Coffee*	`dbpedia/resource/Coffee` *rdf:type* `dbpedia/ontology/Beverage`	R1
B	*Java*	`dbpedia/resource/Java_coffee` *skos:subject* `dbpedia/resource/Category:Coffee`	R1

B	*Java*	`dbpedia/resource/Java_(prog_lang)`
		skos:subject
		`dbpedia/resource/Category:Object-oriented_prog_lang`
			R2

C	*Language*	`dbpedia/resources/prog_lang`
		skos:subject
		`dbpedia/resource/Category:Computer_languages`
			R2

promising. However, this approach assumes that users always use ambiguous tags with just one meaning, which might not be true in some cases.

2.3.6.4 Passant's approach

Passant describes a collaborative approach, where users can manually perform all the tasks in our unified process. These users are assumed to be the taggers in the folksonomy, and will share the results of associating semantics to tags. Hence unlike the approaches above, Passant's approach aims to generate tag-semantics at tag-creation time (Passant, 2007). The data source selected by Passant for the evaluation of this approach is a folksonomy from a corporate Web blog platform, where blog posts are annotated with tags.

Data Selection and Cleaning: In this approach, the data selection and cleaning activity is carried out by each user of the system who can annotate posts. Those tags used in a post that do not have a semantic association are displayed in a different color, so that the contributing user can enrich them semantically. The author does not provide statistics about the data set used to test de approach.

Context identification, disambiguation and semantic identification: These activities are also carried out by users. The assumption is that users know the context because they know what the post content is, and thus if the tag is ambiguous they are able to choose the right meaning, which is defined by concepts or instances of a predefined domain ontology. Polysemic tags can be associated to more than one ontology concept. In this case, users need to associate blog posts with tags, and with the concept they represent to avoid ambiguity. Users are provided with a list of URIs to select from, which are in turn selected from existing ontologies based on how similar concept names are to the given tags.

For our sample folksonomy, let us suppose that a user has two blog posts R1 and R2 in the system. In R1 the user has used two tags *Coffee* and *Java*. Then, he associates each tag with an appropriate class in the preference domain ontology. The system internally associates both tags with the post and their meaning in that post, that is the ontology classes. With respect to R2, the user uses the *Java* tag, among others. Then, he associates this tag with an ontology class in the programming languages domain ontology. In this case, the system associates the *Java* tag, with the post and with its meaning in this post. Therefore, the system is able to differentiate between the two meanings of the *Java* tag, and also the system knows which meaning has been used in which post.

Involving users in the process is a straightforward approach to get rid of ambiguity. However, tagging proved highly successful because of its simplicity of creation. Passant's approach has been tested in a controlled environment of a corporate blog platform, but it has not been evaluated in an open environment. It is unclear how taggers would react to this approach, which controls and restricts their tagging activities. This approach will not scale very well because of its dependence on users to do most of the work themselves.

2.3.6.5 Maala et al.'s approach

The approach of Maala and colleagues uses semantic resources to automatically convert tags of photos into RDF semantic descriptions (Maala et al., 2008). The data source they used in their approach consists of a set of photos and their tags from Flickr.

Data Selection and Cleaning: In this approach, photos with tags that include at least a verb were selected. These tags are then transformed into their non inflectional form using a stemmer. This selection is carried out because the authors needed to semantically describe photos with actions, and in consequence at least a verb is needed. Authors do not provide statistics about the data set used to test the approaches.

Context Identification: Tag context is taken here to be the set of tags used to annotate a particular photo, and thus each photo is processed one at a time. Furthermore, each tag is classified according to one of the following categories: location, time, event, people, camera, and activity. This classification is carried out using some semantic resources including domain ontologies that have been created from external resources such as WordNet and existing Web sites.

Disambiguation: There are no disambiguation activities defined in this approach.

Semantic Identification: The semantic identification activity in this approach creates RDF descriptions for each photo as follows. First, location tags are ordered according to an inclusion relation, and then RDF triples are created stating that the photo is *in* the smallest location, and that this location is *in* a broader location, and so on. Secondly, all 'time' tags are ordered according to an inclusion relation, and then RDF triples are created stating that the photo is *at* the smallest time tag, and that this time tag is *at* a broader time tag, and so on. Thirdly, for each event tag an RDF triple is created stating that the photo *event* is the current event tag. Fourthly, for each camera tag an RDF triple is created stating that the photo was *shot by* the current camera tag. And finally, for each activity tag an RDF triple is created stating that the photo *describes* an activity. Furthermore, WordNet is used to find the arguments of the verb related to the activity including the subject type of who performs the activity. If any of the photo tags correspond to the subject type, then a triple is created stating that the activity *agent* is that tag.

In the context of our sample folksonomy, let us suppose that we have a photo R1, and we have tagged it with all the tags, except for the *Coffee* tag. In addition, we have annotated this picture with more tags including *Madrid*, *Spain*, *January*, *2009*, and *John*. According to this approach, the tags *Madrid* and *Spain* are identified as locations. Hence, we can assert that the photo is *in Madrid*, and that *Madrid* is *in Spain*. Furthermore, the tags *January* and *2009* are identified as time tags, and thus, we can assert that the photo is *at January*, and that *January* is *at 2009*.

On the other hand, the *Program* tag is the unique tag that can be considered as a verb (although it could also be a noun). Therefore, we can assert that the photo *describes* a *Program*. Furthermore, we extract from WordNet the sentence frames of the *Program* verb: 1) *somebody programs something* and 2) *somebody programs*. From these two sentence frames we could identify the arguments of the verb consisting of a mandatory subject of type *somebody*, and of an optional object of type *something*. In this case, the tag *John* could be identified as an instance of *somebody* so that we can assert that *John* is an *agent* of *Program*.

Although the authors of this approach show some examples of use, they do not provide any evaluation metric about the generated RDF descriptions. In a study about photo tags in Flickr the authors estimate that about 53 percent of photos include a tag representing an activity, and thus, the approach leave out of the process the remaining 47 percent of photos.

In the context activity where tags are placed in some predefined categories, some of these tags can be misclassified because of tag ambiguity and the approach does not provide any technique to fix this problem.

2.3.7 Hybrid approaches

So far we have described approaches aiming to group related tags using statistical techniques and some others aiming to associate tags to ontologies. In this section we present some approaches relying on ontologies and clustering techniques whose goal is either to group related tags (Giannakidou et al., 2008) or to associate semantic entities to tags (Specia and Motta, 2007).

2.3.7.1 Giannakidou et al.'s approach

Giannakidou and colleagues proposed a statistical approach for discovering the semantic of tags by clustering tags and resources, being resources represented by their annotations (Giannakidou et al., 2008). This approach is based on a similarity measure that mixes tag co-occurrence with semantic similarity. The approach was tested with a set of Flickr photos depicting cityscape, seaside, mountain, roadside, landscape, sport-scenes and locations.

Data Selection and Cleaning: This approach performs tag spelling normalization where the different spellings of a tag are mapped to a normalized version of that tag. Infrequent tags are filtered out, along with those that do not have a corresponding concept in WordNet, which is the terminological resource they use for calculating semantic similarity. The data set used to test the approach contains 3000 resources. From these resources the 30 most frequent tags were extracted to be analyzed.

Context Identification: Giannakidou and colleagues consider the context for a tag to be the set of tags that co-occur with the given tag when anotating resources. Furthermore, the context of a resource is defined as the tags the users have assigned to it.

Disambiguation: This approach does not explicitly deal with disambiguation problems but authors claim that the grouping of resources and tags found in the next activity helps to disambiguate the meaning of tags.

Semantic Identification: The semantic identification activity creates a graph where the resources, and the most frequent tags in the folksonomy, are represented as vertices. The graph edges associate resources with tags. An edge between a resource and a tag exists if their similarity value is above a certain threshold. In this approach each resource is represented by the set of tags used to annotate it so that the similarity between a tag and a resource is calculated as the maximum similarity value of the tag with each one of the tags used to annotate the resource. The similarity between two tags is a weitghed sum of their social similarity and their semantic similarity. Social similarity is based on the co-ocurrence of both tags when annotating resources. For the semantic similarity authors propose to map tags to concepts in a semantic resource. Then, the semantic similarity is calculated proportionally to the path distance between those concepts in the semantic resource. The bipartite graph relating resources and tags is then clustered using a spectral graph clustering algorithm whose goal is to create disjoint clusters so that the elements in the same cluster have high similarity and elements in different clusters have low similarity.

Let us suppose that the *language* tag has been also assigned to R1 and R3 in the example folksonomy. We have extracted from our folksonomy example the bipartite graph shown in Figure 2.10. This graph relates tags and resources by means of the similiarity value calculated relying on social and semantic similarity. Besides in the bottom of Fig. 2.10 two groups found by the clustering algorithm are shown where the sum of similarities between elements of the same cluster is maximized while the sum of similarities of different clusters is minimized.

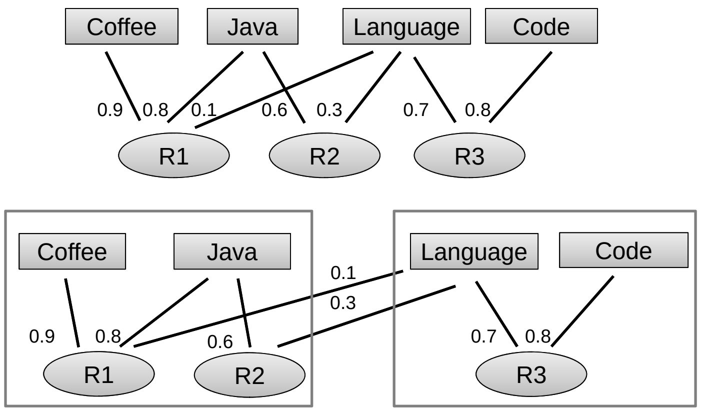


Figure 2.10: Tag clusters generated by Giannakidou *et al.*'s approach. Top: Sample bipartite graph relating resources and tags using social and semantic similarity. Bottom: Clustered graph

Giannakidou and colleagues tested their approach with varying weights assigned to the social and semantic similarity measures to see how each one of these measures affect the clusters found. Authors concluded that social similarity helps to disambiguate ambiguous tags since the context (i.e co-occurring tags) helps to highlight the meaning of tags. Authors also stated that semantic similarity allows to find groups of synonyms but fails to handle ambiguous tags. However, this approach clusters tags into disjoints groups. This means that a tag can belong to just one group and therefore if a tag has several meanings the approach will only identify the most frequent meaning for that tag according to the tag co-occurrence pattern. Moreover, the tags are grouped according to an abstract relation found by the clustering algorithm but this relation is not defined semantically in terms of *subclass*, *part of*, *synonym* or any other relation. Similarly, the meaning of the tags is not defined explicitly.

2.3.7.2 Specia and Motta's approach

Specia and Motta propose a semi-automatic approach using a mix of clustering and ontology-based techniques (Specia and Motta, 2007), focusing on two data sources

(Flickr and Delicious), although the approach could be extended to any other folksonomy data source.

Data Selection and Cleaning: The data selection and cleaning activity starts by filtering out unusual tags. For instance, it filters out tags that do not start with a letter. Then, morphologically-similar tags are grouped using the Levenshtein similarity metric. Finally, infrequent and isolated tags are filtered out. The data set used to test the approach comprised data from Delicious and Flickr. The Delicious data contains 7,164 users, 14,211 resources, and 11,960 tags. On the other hand, the Flickr data consists of 6,140 users, 49,087 resources, and 17,956 tags.

Context Identification: The goal of the context identification activity here is to build clusters of related tags. First, the context of a tag is defined as the set of tags that co-occur with the current tag when annotating a resource or when they are used by the same user. To represent the context of a tag the authors use a vector whose number of elements is equal to the number of distinct tags in the folksonomy, and the values of each position correspond to the number of times the tag co-occurs with the tag corresponding to the current position. In the case where the element of the vector correspond to the tag that is identifying the vector, the value for that element is the frequency of use of that tag in the folksonomy. Then, each tag is compared with other tags using their context vectors in order to find similar tags.

Disambiguation: When a tag is ambiguous it can have more than one pattern of co-occurrence. Thus, the set of similar tags found in the context identification may include tags with different meanings. The disambiguation activity analyzes each group of similar tags in order to find clusters of related tags based on high co-occurrence.

Semantic Identification: Finally, for each cluster of related tags the semantic identification activity is carried out manually. A user uses a semantic Web search engine (e.g, Swoogle[1]) to look for ontologies containing pairs of tags in the cluster. If an ontology is found that contains a pair of tags, then the semantic information about the tags (type, parents, domain, range) is used to establish relations between them.

The proposed approach is depicted in Figure 2.11. Let us apply this approach to our folksonomy example. First, we need to create the vectors to represent each tag and its

[1]http://swoogle.umbc.edu/

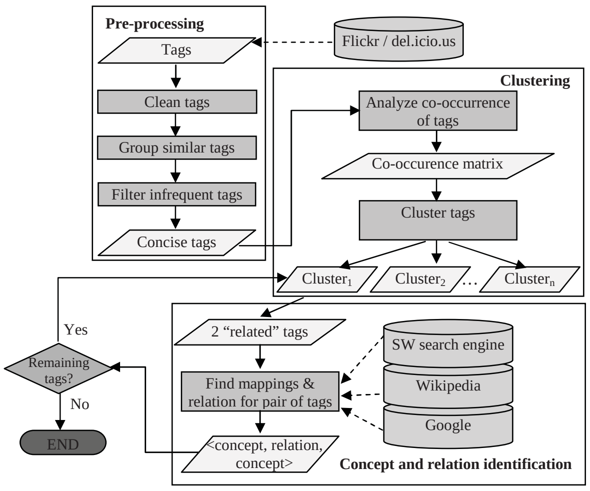


Figure 2.11: Process of associating semantics to tags (Specia and Motta, 2007)

context. Some examples of these vectors are shown in table 2.7. Then, over this matrix we have to apply the clustering algorithm proposed in the disambiguation activity. As a result of this activity we can get two two groups of tags. A group with tags *Coffee* and *Java*, and the other group with tags *Language*, *Java*, *Program* and *Code*. Then, the user looks in Swoogle for each pair of tags in each group and tries to establish manually the relations among tags. The Cyc Ontology[1] has a direct relation among the tags *coffee* as a beverage and *Java*, being the latter an *english alias* of the former. On the other hand, in the LT4eL ontology[2] we found that *Java* is a *subclass of Language* as a programming language. In addition, we found that *Code* as source code is *subclass of Program* as a computer program. The final ontology is depicted in Figure 2.12.

Table 2.7: Tag and context vector representation

	Coffee	*Java*	*language*	*Program*	*Code*
Coffee	1	1	0	0	0
Java	1	5	3	2	0
Language	0	3	3	2	0

[1]`http://sw.opencyc.org/`

[2]`http://www.lt4el.eu/index.php?content=tools#ontology`

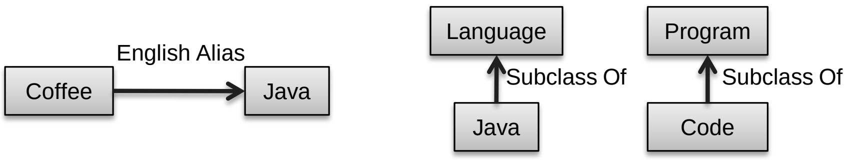


Figure 2.12: Tag relations identified within group of tags

One of the main advantages of this approach is that it combines clustering and ontology-based techniques. However, the approach has also some limitations. For example, the semantic identification activity requires users to analyze manually the ontologies retrieved from a Semantic Web search engine like Swoogle. However, an approach to automate this process has been introduced in Angeletou et al. (2008). Specia and Motta's approach is highly dependent on finding relations between tags in existing ontologies. It is therefore natural to expect that many tag pairs found in folksonomies will not be found in any ontology libraries, thus limiting the output of this approach.

With respect to evaluation, the authors do not evaluate how well the clusters of highly co-occurring tags in the similar tag sets help in the disambiguation of tag senses.

2.3.8 Consolidated overview of approaches

In this section we will provide a summary and comparison of the approaches. This overview is summarized in Table 2.8, which uses the activities identified in our unified process and some other characteristics that we have considered in our descriptions. In this table, the first column contains the reviewed approaches. The following columns are the characteristics evaluated using the unified process. The approach type can take the values of *Stat* for Statistical-based approaches, *Ont* for ontology-based approaches, and *Hyb* for hybrid approaches. The Auto column describes if the approach is automatic or manual, and it can take the values of *Yes* for automatic, *No* for manual or *Semi* for semiautomatic. The data source column shows the folksonomies used to test the approaches in the original publications. The acronyms used are *Del* for Delicious, *Pol* for Polyphonet, *Bib* for Bibsonomy, *Raw* for Rawsugar, *Fli* for Flicker, and *Oth* for other data sources. Then, there are columns to specify if the approaches include or not some of the activities of the proposed general process: data selection and cleaning, context identification, disambiguation and semantic identification.

In addition, the ninth column identifies the output type of the approach which can be an ontology (Onto), a hierarchy of tags, clusters of tags, instances of ontology classes, or enrichments (*i.e.*, tags are related to semantic entities). The tenth column states, for the cases where the output is an ontology, whether the knowledge in the ontology has been properly evaluated. By properly we mean assessing the precision and recall of the classes and relations within the ontology with respect to a given scope, as well as other ontology quality measures such as modularity. From the review of the approaches we have identified different types of evaluations performed by their authors which includes: 1) human-based comparison of the generated ontologies (among them) in a given domain, 2) descriptive analysis of a small part of the output, 3) the output is included in a process which is then evaluated (*e.g.*, recommendation systems, or search engines), and 4) the use of metrics such as precision and recall to evaluate the output.

Eight of the approaches use statistical techniques to identify the hidden semantics of tags in folksonomies, while five more use ontologies to associate semantics to tags, and just two use a hybrid approach. Statistical techniques are used most of the time to find groups of related tags and hierarchies, whereas ontology-based approaches are used to associate semantic entities to individual tags.

Most of the approaches are automatic, except for Specia and Motta (2007) and Limpens et al. (2010), which is semiautomatic, and Passant (2007) which is completely manual and focuses on user-generated semantic enrichment. The most studied data sources are Delicious and Flickr. In Hamasaki et al. (2007), the folksonomy was adapted from an academic conference support system, and in Passant (2007) a folksonomy of an enterprise blogging platform was used. Almost all the approaches implement a data selection and cleaning activity, defining the initial tag set and filtering out the tags they do not want to deal with, except for Benz et al. (2010); Hamasaki et al. (2007); Limpens et al. (2010) and Heymann and Garcia-Molina (2006) where this activity was not described.

In all approaches some kind of context identification is included. The objective of this activity is usually for tag disambiguation or for semantic identification. Table 2.9 presents the different context definitions found in the reviewed approaches. Most of the approaches rely on tag co-occurrence when annotating resources regardless

Table 2.8: State of the art: consolidated overview

Approach	**Type**	**Auto**	**Data Src**	**Select. Clean.**	**Context Ident.**	**Disambi-- guation**	**Sem. Ident.**	**Output**	**Onto. Eval.**	**Dom. Know.**
Mika, 2007	Stat	Yes	Del,Oth	Yes	Yes	No	Yes*	Onto.	No^{1,2}	No
Hamasaki *et al.*, 2007	Stat	Yes	Pol	No	Yes	Yes	No$^{\mp}$	Onto.	No^{2,3}	No
Jäschke *et al.*, 2008	Stat	Yes	Del,Bib	Yes	Yes	No	No$^{\mp}$	Hier.	-^{2}	No
Limpens *et al.*, 2010	Stat	Semi	Oth	No	No	Yes	Yes*	Enri.	-^{4}	No
Begelman *et al.*, 2006	Stat	Yes	Del,Raw	Yes	Yes	No	No$^{\mp}$	Clus.	-^{2}	No
Kennedy *et al.*, 2007	Stat	Yes	Fli	Yes	Yes	Yes	Yes*	Inst.	-^{3,4}	No
Heymann & Garcia Molina, 2006	Stat	Yes	Del,Cit	No	Yes	No	No$^{\mp}$	Hier.	-^{3}	No
Benz *et al.*, 2010	Stat	Yes	Del	No	Yes	Yes	Yes*	Hier.	-^{4}	No
Giannakidou *et al.*, 2008	Hyb	Yes	Fli	Yes	Yes	Yes	No$^{\mp}$	Clus.	-	No
Specia & Motta., 2007	Hyb	Semi	Del,Fli	Yes	Yes	Yes	Yes	Onto.	No^{2}	No
Angeletou *et al.*, 2008	Ont	Yes	Fli	Yes	Yes	Yes	Yes^{+}	Enri.	-^{4}	No
Cantador *et al.*, 2008	Ont	Yes	Fli,Del	Yes	Yes	No	Yes^{+}	Inst.	-^{3,4}	No
Tesconi *et al.*, 2008	Ont	Yes	Del	Yes	Yes	Yes	Yes^{+}	Enri.	-^{4}	No
Passant, 2007	Ont	No	Oth	Yes	Yes	Yes	Yes^{+}	Enri.	-^{2}	No
Maala *et al.*, 2008.	Ont	Yes	Fli	Yes	Yes	No	Yes	Enri.	-^{2}	No

Superindexes: * Semantics limited to few relations, + tags are related to semantic resources, ∓ groups of related tags,
1 comparison of the generated ontologies among them, 2 descriptive analysis of a part of the evaluated data, 3 Task-based evaluation,
4 Accuracy/Precission-Coverage/Recall metrics for the evaluation

of the user. However, as these approaches ignore the user in the context definition, they are mixing the different meanings of tags given by the different users. Other approaches, such as Maala et al. (2008) analyzes the tags in the user post getting rid of the use of tags in different meanings. Tesconi et al. (2008) propose to use the co-occurring tags in the resources annotated by the user. If a user has used the tag in more than one sense, then the analysis of this context will result in the most frequent sense for that tag. The idea behind using the co-occurring tags in the user social network first introduced in Hamasaki et al. (2007), is that groups of people sharing some interest, e.g, scientist and practitioners, tend to use the same vocabulary in a particular field. Finally, only Kennedy et al. (2007) use tagging metadata information as part of context definition.

Table 2.9: Context definitions

Approach	**Context**
Maala *et al.*, 2008 García-Silva *et al.*, 2009	Co-occurring tags in the user post
Tesconi *et al.*, 2008	Co-occurring tags when annotating resources tagged by the user
Hamasaki *et al.*, 2007 García-Silva *et al.*, 2009	Co-occurring tags in the user social network
Mika, 2007 Giannakidou *et al.*, 2008 Specia and Motta, 2007	Co-occurring tags when they are used to annotate a resource
Mika, 2007 Specia and Motta, 2007	Co-occurring tags when they are used by an annotator
Angeletou *et al.*, 2008	All analyzed tags
Kennedy *et al.*, 2007	Latitude and longitude of the geo-tagged resources and timestamp
Cantador *et al.*, 2008	Wikipedia category list associate to each tag

Six of the research works ignore the disambiguation problem, while the other ninth suggest some technique to disambiguate tag meanings. Regarding the semantic identification activity Hamasaki et al. (2007), Jäschke et al. (2008), Begelman et al. (2006), Heymann and Garcia-Molina (2006), and Giannakidou et al. (2008) do not describe how to define explicitly or formally the tag semantics.

Mika (2007) uses social network metrics and set theory to define a limited semantics of the relations between tags. Limpens et al. (2010) identify spelling variations and homonyms using string similarity metrics and other metrics based on the folksonomy structure. Benz et al. (2010) generate a tag hierarchy based on more general and more specific terms, and provide sets of synonyms for each tag in the hierarchy. On the other hand Angeletou et al. (2008); Passant (2007); Specia and Motta (2007); Tesconi et al. (2008) use external semantic resources. Maala et al. (2008) also use external semantic resources to define in which category (e.g, location, time, event, etc.), each tag fits better, but in the end the RDF triples generated are not related to the semantic resources, losing the advantage of using the obtained intermediate results. Similarly Kennedy et al. (2007) categorize tags in two classes, locations and events, though they do not use external resources but metadata information.

Passant (2007) approach differs from all others in that all the activities are carried out manually by users. For instance, the user is the one who decides which tags are to be enriched semantically. Also, when the user has to define which ontology concept he wants to associate with the tag, he has to understand the context in which the tag is used and if the tag is ambiguous then he has to define the right meaning in order to associate the best ontology concept. Furthermore, the user will need to understand the meaning of the ontology concepts the system suggests for the given tags to be able to select the correct URIs.

From all the approaches only three claim that they generate ontologies (Hamasaki et al., 2007; Mika, 2007; Specia and Motta, 2007). In fact Hamasaki et al. (2007) do not provide semantics to the identified group of tags. Mika (2007) proposes some techniques to identify broader or narrower relations, though in this work the obtained relations are exemplified, but they are not evaluated. In the case of Specia and Motta (2007) the semantic entities and relations, which have been manually found in existing ontologies, are not limited to a given scope nor they are evaluated. Kennedy et al. (2007) and Cantador et al. (2008) classify tags under classes, Heymann and Garcia-Molina (2006); Jäschke et al. (2008) and Benz et al. (2010) generate hierarchy of tags, Begelman et al. (2006) and Giannakidou et al. (2008) cluster tags, and Angeletou et al. (2008); Maala et al. (2008); Passant (2007); Tesconi et al. (2008) and Limpens et al. (2010) enrich tags with semantic entities.

Finally we want to note that none of the approaches aiming at generating an ontology narrow the scope of the ontology to a given domain.

2.3.9 Conclusions

In spite of the amount of research works aiming at generating semantic structures from folksonomies (*i.e.*, tags networks, clusters, hierarchies, and enrichments) the objective of obtaining a formal ontology is still a current challenge. For us a formal ontology consists of concepts and well defined relations among them. Most of the efforts in this respect have elicited a graph of tags, and in some cases a limited semantics about tag relations has been suggested. In addition most of the approaches do not differentiate the ontology schema (*i.e.*, concepts and relations) from the facts (*i.e.*, instances). Moreover none of these approaches have focused on limiting the scoped of these semantic structures to a given domain. Therefore we can state the following open research problem:

- There is a lack of a method to automatically elicit domain knowledge from folksonomies that can be used in the development of formal domain ontologies.

2.4 Methodologies for building ontologies

The development of ontologies has evolved from an initial stage where authors follow their own set of principles and design rules when developing ontologies, to a more advance stage where there are available methodologies for building ontologies. These methodologies provide definitions of ontology development processes and ontology life cycles where these processes are organized. In this section we review four well-known methodologies for building ontologies from scratch (Methontology, On-to-Knowledge, and DILIGENT) and a novel methodology (NeOn) that gives support to collaborative processes for building ontology networks. The NeOn methodology has identified several possible scenarios for building ontologies, including an scenario where it is possible to reuse knowledge from non-ontological resources. Non-ontological resources (Villazón-Terrazas, 2012) are "knowledge resources whose semantics have not been yet formalized explicitly by means of ontologies". This definition of non-ontological resource covers user-generated classification systems such as Folksonomies since it has been shown that it is possible to consider these systems as sources of knowledge (Begelman et al., 2006; Cattuto et al., 2008; Marlow et al., 2006). The objective of this review of methodologies for building ontologies is to show that existing methodologies lack methods and techniques to develop ontologies from user-generated classification systems.

2.4.1 METHONTOLOGY

METHONTOLOGY (Fernández-Lopez et al., 1997) covers distinct activities of the ontology development process, an ontology development life cycle, and techniques to carry out the activities related to the development, support and management of the process.

The ontology life cycle is based on evolutionary prototypes which allow refining the ontology by means of the addition, deletion, or change of terms in the ontology. For each prototype the development process consists of the following activities: ontology specification, conceptualization, formalization, implementation and maintenance. In addition METHONTOLOGY defines support activities such as knowledge acquisition, integration, merging and alignment, evaluation, documentation, and configuration management. Finally the methodology describes management activities including scheduling, control and quality assurance.

Note that in this methodology it is possible to reuse existing ontologies since METHONTOLOGY includes the support activities of integration and, merging and alignment. Nevertheless, authors did not provide any activity for the knowledge acquisition from non ontological resources.

2.4.2 On-To-Knowledge

The objective of the On-To-Knowledge methodology (Sure et al., 2004) is to develop ontologies fitted to the requirement of knowledge management systems, and thus these ontologies are highly dependent of the application where they are going to be used. On-To-knowledge proposes the following processes: feasibility study, ontology kick off, refinement, evaluation, and maintenance. Moreover On-To-Knowledge also describes a cyclic ontology life cycle, which is based on evolutionary prototyping.

In this methodology authors propose to identify, in the kick off process, existing ontologies to be reused in the development process. In addition, the methodology suggests, in the refinement process, to apply ontology learning approaches to reduce the efforts required to develop new ontologies. Nevertheless ontology learning, at this time, was limited to knowledge acquisition from text documents.

2.4.3 DILIGENT

DILIGENT (Pinto et al., 2004) is a methodology for building ontologies in distributive and collaborative settings. This methodology proposes the following development phases: build, local adaptation, analysis, revision, and local update. These phases are arranged in an ontology life cycle model based on evolutionary prototyping. DILIGENT can be described as an argumentation framework where the different stakeholders of the development process can discuss about the changes proposed for the ontology in the different phases of the life cycle. This methodology does not propose to reuse existing ontologies or non ontological resources in the ontology development process.

2.4.4 NeOn

The NeOn methodology (Suarez-Figueroa, 2010) is a scenario-based methodology for building ontology networks in collaborative environments. It puts special emphasis on the reuse and reengineering of knowledge resources. NeOn methodology includes an ontology development process, life cycle models and techniques and tools to support the ontology development process. This methodology proposes nine different scenarios for building ontologies:

- Scenario 1: From specification to implementation.
- Scenario 2: Reusing and re-engineering non-ontological resources.
- Scenario 3: Reusing ontological resources.
- Scenario 4: Reusing and re-engineering ontological resources.
- Scenario 5: Reusing and merging ontological resources.
- Scenario 6: Reusing, merging and re-engineering ontological resources.
- Scenario 7: Reusing ontology design patterns.
- Scenario 8: Restructuring ontological resources.
- Scenario 9: Localizing ontological resources.

Figure 2.13 depicts the different scenarios and the main activities proposed for each of them. All these scenarios can be mixed during the development of an ontology network. The first scenario is mandatory since it contains the core of activities for building ontologies: specification, scheduling, localization, restructuring, conceptualization, formalization, and implementation. The development of ontologies from non-ontological resources is covered by the scenario 2. This scenario describes the cases where non-ontological resources have to be re-engineered to turn them in ontologies.

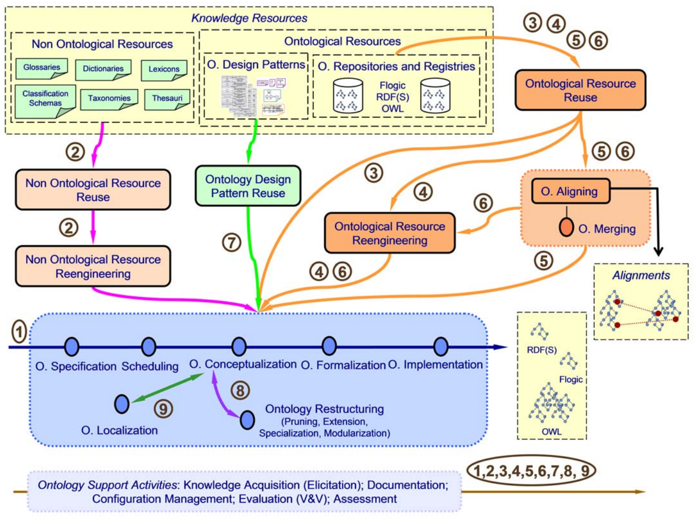

Knowledge Resources
Non Ontological Resources
Glossaries
Dictionaries
Lexicons
Classification Schemas
Taxonomies
Thesauri
Ontological Resources
O. Design Patterns
O. Repositories and Registries
Flogic
RDF(S)
OWL
Ontological Resource Reuse
Non Ontological Resource Reuse
Ontology Design Pattern Reuse
Non Ontological Resource Reengineering
Ontological Resource Reengineering
O. Aligning
O. Merging
Alignments
O. Specification
Scheduling
O. Conceptualization
O. Formalization
O. Implementation
O. Localization
Ontology Restructuring (Pruning, Extension, Specialization, Modularization)
Ontology Support Activities: Knowledge Acquisition (Elicitation); Documentation; Configuration Management; Evaluation (V&V); Assessment
1,2,3,4,5,6,7,8, 9

Figure 2.13: Scenarios for building ontologies in the NeOn methodology

In (Villazón-Terrazas, 2012) the author proposes a complete set of methodological guidelines, patterns and tools to support the reuse and reengineering of non-ontological resources in ontologies. The non-ontological resources addressed in this work are: classification schemes, thesauri and lexica. In addition Villazón-Terrazas (2012) identify user-generated classification systems as resources from which it is possible to elicit knowledge. However in this work the author does not propose methods and techniques for the specific purpose of developing ontologies from user-generated classification systems.

2.4.5 Conclusions

In this section we surveyed four of the main methodologies for building ontologies. We have seen that METHONTOLOGY and DILIGENT do not include in the ontology development process the possibility of reusing non-ontological resources. In contrast, On-To-Knowledge considers the use of ontology learning approaches though these approaches are limited to text documents. From all the surveyed methodologies NeOn is the methodology that put emphasis on the reuse of non-ontological resources. In fact, NeOn methodology describes the scenario 2 for building ontologies by reusing and reengineering non-ontological resources. This scenario has been addressed in detail in (Villazón-Terrazas, 2012) where the author has presented methods, patterns and tools for building ontologies from classification schemes, thesauri and lexica. However in this work user-generated classification systems are not addressed.

Therefore we can conclude that none of the aforementioned methodologies propose methods and techniques to tap into user-generated classification systems for building ontologies.

2.5 Conclusions

We started this chapter presenting named lists as an example of user-generated classification systems in the current web. We concluded through an example that named lists seem to be an interesting emergent classification system from which we can obtain knowledge. However these lists have not been studied as source of emergent semantics that can be used in knowledge acquisition processes. Thus we defined the first research problem to be addressed in this thesis: **Is it possible to elicit knowledge, in the form of semantically related terms and explicit relations between them, from the classification systems emerging from named lists?**

Next, we review the different attempts to elicit semantic structures from folksonomies. These approaches range from those grouping and relating tags, according to tag similarity measures, in the hope that such grouping exposes the tag meanings, to approaches where tags are associated with existing semantic entities in knowledge bases so that their meaning can be asserted. The semantic structures produced by the reviewed approaches include ontologies, tag clusters, tag hierarchies, and tag enrichments. Despite the high number of works reviewed we showed that just a fraction of them produce formal ontologies. We also showed that most of the works defines a very limited semantics of the relations within the produced ontologies. Finally we noted that none of these research works define the ontology scope in a given domain. Thus we defined as an open research problem **the lack of a method to automatically elicit domain knowledge from folksonomies that can be used in the development of formal domain ontologies**.

Finally we presented the most representative methodologies for building ontologies and showed that just On-To-Knowledge and NeOn methodology consider reusing existing knowledge. However On-To-Knowledge is limited to ontology learning approaches from text documents. On the other hand, the NeOn methodology mentions the reuse of user-generated classification systems in its scenario number 2 *building ontologies by reusing and reengineering non-ontological resources*. However, this methodology does not include a method or techniques to leverage the knowledge of user-generated classification systems. Therefore we stated that **current methodologies for building ontologies lack methods and techniques for building ontologies from user-generated classification systems**.

CHAPTER 3

OBJECTIVES AND CONTRIBUTIONS

In this chapter we describe the objectives pursued by this thesis together with its main contributions. In addition we present the open research problems we are addressing along with the restrictions and assumptions upon which our research relies. Finally we describe the research methodology that we followed during the development of this thesis.

The global objective of this thesis is **to investigate methods and techniques to tap into user-generated classification systems and the web of data to develop ontologies**. Thus, this research is framed within the ontology development scenario (Suárez-Figueroa et al., 2012) where ontology developers want to reuse and re-engineer existing knowledge sources, which has not been formalized yet, in the ontology development process (Villazón-Terrazas, 2012). In this context we list the main contributions of this thesis.

i) An integrated method to create ontologies from user-generated classification systems. This method covers different stages of the ontology development process:

   - Elicitation of domain terminology by collecting relevant terms from user-generated classification systems.
   - Identification of classes from the extracted terminology by reusing existing classes of ontologies in the web of data.
   - Discovery of relations between the identified classes by reusing existing relations of ontologies in the web of data.

ii) We have identified useful techniques for implementing some of the activities in the method and propose how to adapt them according to the requirements of each of these activities. We have used the following techniques in our method:

   - Spreading activation (Crestani, 1997) for eliciting domain terminology. We have applied this technique to traverse a network of category names which has been extracted from user-generated classification systems, so that we can collect relevant domain terms from it.
   - Vector space model (Salton and Mcgill, 1986) for grounding category names to semantic entities. We have applied this technique to select from a knowledge base the semantic entity that better represents the meaning of a category name in a given context.
   - Dynamic queries (Heim et al., 2010) in SPARQL (Prud'hommeaux and Seaborne, 2008) for identifying classes. We have used this technique to pose SPARQL queries to traverse all the possible paths in an RDF graph, linking through *same as* relations a semantic entity and a class, so that we can identify which semantic entities are considered as classes in the corresponding knowledge base.
   - Dynamic queries in SPARQL for discovering relations. We have used this technique to pose SPARQL queries to traverse all the possible paths in an RDF graph, linking two classes so that we can identify in the corresponding knowledge base the relationships between a pair of classes.

iii) A study about the emerging semantics in the user-generated classification system defined by Twitter Lists. In this study we established relatedness measures between category names based on the distinct user roles interacting with the lists. Then we compare these relatedness values with the results of well-known relatedness measures based on WordNet. We also included a comparison with relations found in DBpedia (Bizer et al., 2009b) and some other linked data sets. Our conclusions included the name of the relations found and the amount of them in the data set used for the experiments.

3.1 Objectives

The main objective of this thesis is to propose a method, and identify and adapt techniques to automatically develop domain ontologies leveraging folksonomies and ontologies in the web of data. This objective can be broken in the following detailed objectives:

O1. To propose methods and techniques to leverage the knowledge in user-generated classification systems in ontology development.

O2. To propose methods and techniques to reuse ontologies in the web of data within the development process of a new ontology.

O3. To propose an integrated method to create ontologies relying on user-generated classification systems and ontologies in the web of data.

O4. To analyse the emerging semantics of twitter lists, a user-generated classification system which has not been studied as source of knowledge yet.

3.2 Contributions to the State of the Art

We have contributed with a new method, the identification of techniques supporting the method, and surveys to the two open research problems identified in the state of the art chapter (see chapter 2). In this section we present the contributions generated for each of these problems.

I. With regard to the *lack of methods and techniques to elicit domain ontologies from folksonomies and named lists* we have generated the following contributions:

   C1. **An integrated method to develop ontologies from user-generated classification systems**. This method, presented in chapter 4, comprises two processes and defines how different user roles participate in each process. To present the method and how the processes are arranged in a workflow we use the Data-Intensive Systems Process Engineering Language DISPEL (Martin and Yaikhom, 2011), which allows a clear description of the method components, their interaction and the data transformation. DISPEL and the framework used to present the method are described in chapter 4.

C2. **Identification and adaptation of a set of techniques to support the method processes**. We present in chapter 5 a set of techniques to automatically carry out the processes defined in the method. These are existing techniques, some of them applied in other research fields such as Information Retrieval, that we have adapted to fulfill the objectives defined for each process.

II. With regard to the *lack of a study of named lists as source of emergent semantics* we have the following contribution:

C3. **A characterisation of the emergent semantics in Twitter lists**. Though folksonomies have drawn the attention of the research community, other classification systems which share many similarities with them, such as named lists, have been ignored as a source of knowledge. We present, in chapter 6, a survey of the emerging semantics in Twitter lists, where we carry out an objective and quantitative analysis of the kind of information that can be extracted from these user-generated lists.

3.3 Assumptions

The method proposed and the techniques used in this thesis rely on the following assumptions:

A1. It is possible to access public data in user-generated classification systems.

A2. User-generated classification systems contain sufficient individual classifications from which it is possible to identify an emerging vocabulary according to frequency of use the category names.

A3. Relevant domain terms are those frequently used in that domain.

A4. User-generated classification systems contains to some extent domain data.

A5. There are publicly available ontologies that cover to some extent domain data.

A6. There are techniques which can be adapted to automatize the tasks identified in the method.

A7. There are similarity measures based on knowledge bases useful for analyzing the emergent semantics in user-generated lists.

3.4 Hypotheses

This thesis work was developed based on the following hypotheses:

H1. User-generated classification systems can be mined to collect knowledge pertinent to a domain so that this knowledge can be used in an ontology development process.

H2. Ontologies in the web of data can be used to make explicit and formal the knowledge extracted from user-created classification systems.

H3. It is possible to create a method, relying on the hypotheses H1 and H2, to develop domain ontologies from user-generated classification systems.

H4. It is possible to adapt existing techniques to automatically carry out the processes proposed in the method.

H5. There is an emergent semantics which result of the aggregation of the individual classifications in named lists.

H6. It is possible to use existing similarity measures based on WordNet to analyze the emergent semantics in named lists.

H7. It is possible to use existing ontologies published in the web of data to analyze the emergent semantics in named lists.

3.5 Limitations

Finally we want to mention the restrictions of the method and techniques developed in this thesis that highlight possible future work.

R1. The data to be extracted from the user-generated classification system is limited to the information of public use.

R2. The ontologies to be reused have to be available locally and stored in a unique repository. Our approach does not reuse ontologies distributed in different repositories.

R3. Domain experts and ontology engineers are involved in the method to provide only input information required by the proposed method.

R4. The knowledge that can be reused and quantified from existing ontologies is limited by the knowledge contained in these ontologies.

R5. The produced ontology is in English language. We only benefit of the category names written in English in the classification systems and we only reuse ontologies in English.

R6. The output ontology does not contain instances.

In table 3.1 we present how the contributions of this thesis work fulfill the objectives that we posed at the beginning of our research. We also specify the assumptions on which the contributions relies and the hypotheses that they validate. In addition, we define the scope of each contribution by means of its associations with the restrictions that we defined in this section.

3.6 Research methodology

This research was inspired by early work in emergent semantics (Aberer et al., 2004), and the increasing interest in collaborative tagging systems from a knowledge acquisition perspective (Mika, 2007; Specia and Motta, 2007). Mika (2007) and Specia and Motta (2007) claimed that it was possible to obtain an emergent semantics from these classification systems and they identified benefits for folksonomy-based systems and for knowledge acquisition processes. Folksonomy-based systems benefits include better recommendation and searching functionalities since these functions can rely on the emergent semantics to improve their results. On the other hand, knowledge acquisition processes benefit from folksonomies since these processes have available a dynamic source of information maintained by potentially large user communities from which knowledge can be extracted automatically.

Table 3.1: Objectives and contributions associated with assumptions, hypotheses and restrictions

Objective	Contribution	Assumptions, Hypotheses, and Restrictions
O1. To leverage the knowledge in user-generated classification systems in the ontology development. O2. To reuse ontologies in the web of data within the ontology development process.	C1. An integrated method to develop a domain ontology from user-generated classification systems and relying on ontologies in the web of data.	<A1,A2,A3,A4,A5>,<H1,H2,H3>, <R1,R2,R3,R4,R5,R6>
O3. To develop ontologies relying on user-generated classification systems and linked data.	C2. An adaptation of a set of techniques to support the distinct activities and tasks proposed in the method	<A5>,<H4>, <R1,R2,R3,R4,R5,R6>
O4. To analyse the emerging semantics of twitter lists, a user-generated classification system which has not been studied as source of knowledge yet.	C3. A survey of the emergent semantics in Twitter lists	<A1,A2,A3,A6>,<H5,H6,H7>, <R1,R2,R4>

Initially we defined a broad research problem: **to survey the potential use of the emergent semantics from folksonomies in innovative knowledge acquisition processes**. To refine this research problem and define the objectives and hypotheses of the thesis we followed an iterative research methodology consisting of two stages (see figure 3.1). In the first stage we used an **exploratory approach** (Kothari, 2004). The objective of exploratory research is to define the research problem and the hypotheses to be tested. Accordingly, in the first stage, we reviewed the state of the art of approaches mixing the topics of folksonomies and semantics to know more about: i) the problems being solved with semantics and folksonomies, ii) the approaches proposed to solve these problems, and iii) the strengths and drawbacks of these approaches. This review of the state of the art, which was presented in chapter 2, helped us to specify in detail the definition of the research problem and the hypotheses of this work.

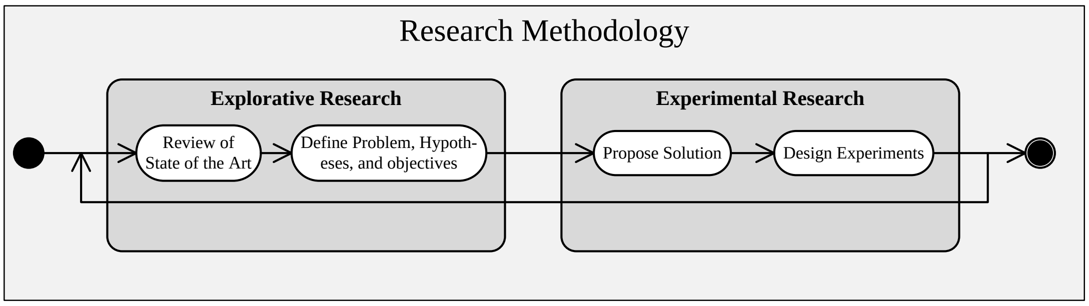


Figure 3.1: Iterative research methodology using exploratory and experimental approaches.

While we discarded problems such as the use of ontologies to ameliorate interoperability issues in folksonomies (Kim et al., 2008b), and the use of the ontologies to improve the functionalities in systems which were based on folksonomies (Angeletou et al., 2009), we defined our research problem more precisely in terms of **leveraging folksonomies in the development of domain ontologies with a well defined semantics**. We emphasized the requirement of a well defined semantics for the ontology since most of the approaches in the state of the art produced ontologies where i) the relations between terms were established but not defined, or in some cases defined to a limited extent, and ii) there were not a clear distinction between classes and instances. The hypotheses on which we rely to propose a solution for this problem were presented in chapter 3, and we remark two of them: i) it is possible to obtain a vocabulary from folksonomies pertinent to a given domain, and ii) it is possible to reuse knowledge in existing ontologies, which have been published under the linked data principles, to formalize the semantics of that vocabulary.

Once we had defined the research problem we proceeded to the second stage where we follow an *experimental approach* (Dodig-crnkovic, 2002; Kothari, 2004). Our objective in the experimental research was to propose a solution based on the hypotheses to fulfill the research objectives, and design experiments to validate the hypotheses. In this stage we investigated existing techniques in other research fields such as graph theory and information retrieval which might help to reach the objectives. Then we adapted these techniques to the requirements defined by the particularities of our research. After this we designed the experiments to validate the proposed solutions. The experiments were carefully designed so that they can be reproduced by third parties. Therefore, we made public the datasets of each experiment, and used well-known evaluation metrics. Next, we carried out an abstraction exercise over the procedure that we had followed when developing the techniques, and designing and executing the experiments. The objective was to elicit commonalities in the form of activities, user roles, and a workflow where these activities were organized. Thus, with these components (activities, user roles and the workflow) we produced the method that we are proposing in this thesis.

During the first iteration we addressed folksonomies as source of knowledge for the ontology development. While we were developing this research we witnessed how some web applications started allowing users to create named lists as a mean to organize and share information. We noticed the commonalities between these named lists and folksonomies and hence we generalized these structures in a conceptualization named user-generated classification systems. Therefore we carried out a second iteration in the research process so that we could study if these named lists could be used in the same manner as folksonomies, that is as a source of knowledge in the ontology development. We carried out a review of the state of the art about the use of named lists as a source of knowledge and we did not find any research work in this respect. Thus, we surveyed the emergent semantics that can be elicited in different ways from named lists. This survey showed that is possible to obtain semantics from these classification systems and therefore can be leveraged in knowledge acquisition processes. Nevertheless, due to time constraints the use of named lists in ontology development has been postponed and it is included in the future work.

CHAPTER 4

A METHOD FOR DEVELOPING ONTOLOGIES FROM USER-GENERATED CLASSIFICATION SYSTEMS

In this chapter we describe our method for developing ontologies from user-generated classification systems. First we present in section 4.1 preliminary information about i) the terminology used in the method description, ii) the user roles participating in the method, iii) the formal language used to present the different components of the method, iv) a model of user-generated classifications systems in this language, and v) an illustrative example which is going to be used throughout this chapter when describing the different parts of the method. Next in section 4.2 and 4.3 we describe the two processes making up the method, which are then organized in a ordered sequence in section 4.4.

4.1 Preliminaries

In this thesis we propose a method, and supporting techniques, to develop domain ontologies from user-generated classification systems in the Web. Since the terms *method* and *technique* are often used as interchangeable words we present the terminology that we are going to use in the rest of the chapter so that we can shed light on their meaning.

4.1.1 Terminology

We follow throughout this document the definitions of method and technique given in the IEEE standards. A method (IEEE, 1990a) is a "set of orderly processes or procedures used in the engineering of a product or performing a service". A technique (IEEE, 1990b) is "a technical and managerial procedure used to achieve a given objective". In figure 4.1 we show that methods consist of processes and are specified with techniques. Further, Greenwood (Greenwood, 1973) claims that a method is a general procedure, whereas a technique is the specific application of a method and the way in which the method is executed. Several techniques are used for applying a given method, and each technique specifies what means should be used to execute the method. Though we are not proposing a methodology, we consider important to note that methodologies consists of methods and techniques. We use these definitions of method and technique to define the ontology development process so that we can distinguish between the conceptual descriptions of the processes from their technical implementations.

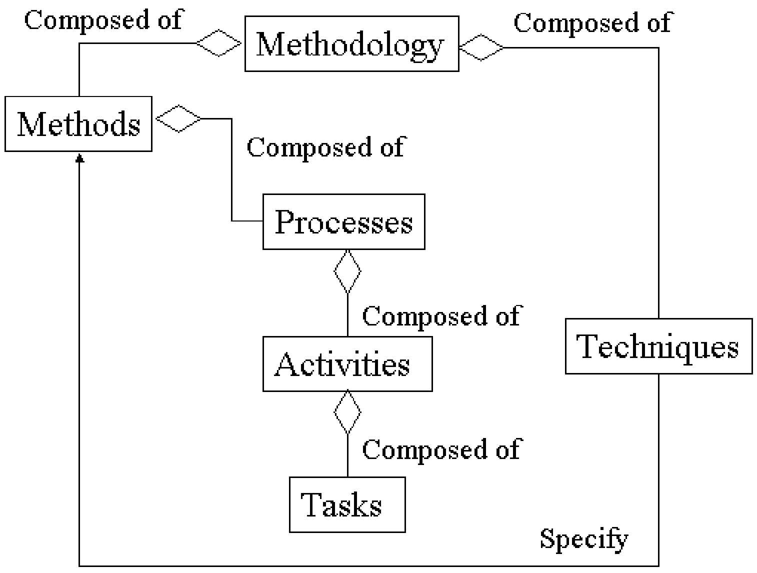


Figure 4.1: Methods and techniques (Gómez-Pérez et al., 2004)

In addition, methods are comprised of processes. According to IEEE (1990b) a process is "a sequence of steps performed for a given purpose, including its required input and output information". Processes consist of activities, which in turn consist of tasks. Therefore the relation between processes, activities and tasks is mereological and it depicts different levels of specificity of the work that has to be executed.

4.1.2 User roles

We define two user roles (or profiles) to distinguish the different skills required to provide input data to some of the processes of the method. Similarly to what happens in ontology development methodologies, such as (Gómez-Pérez et al., 2004; Suárez-Figueroa et al., 2012), and methods (Villazón-Terrazas et al., 2010), we define two user roles: *ontology engineers* and *domain experts*. Ontology engineers have a comprehensive knowledge in methodologies, methods and techniques to develop ontologies, ranging from ontology specification to ontology implementation and maintenance. In addition ontology engineers must have knowledge about software engineering and programming so that they can design and implement scripts and programs that can be helpful when automatizing some processes. On the other hand, domain experts are users with a broad experience in the domain of study. They have to be able to provide models of the domain in a high level abstraction as well as in detailed manner. Domain experts do not require knowledge about ontology development though having these skills may help to get better conceptualizations.

4.1.3 Data-intensive systems process engineering language (DISPEL)

To formalize the method and its processes we use the *Data-Intensive Systems Process Engineering Language* DISPEL (Martin and Yaikhom, 2011), which was developed under the framework of the ADMIRE project[1]. DISPEL[2] is a high-level scripting language used to describe abstract workflows for distributed data-intensive applications. Though there are commercial workflow engines, such as WebSphere MQ Workflow[3] and Oracle Human Workflow [4], following the *Business Process Execution Language* BPEL (Juric, 2006), they are focused on modeling and orchestrating business processes and human actions. In contrast DISPEL is more oriented to controlling computations and managing data movements.

Abstract workflows are well suited for describing methods since they are focused on the description and understandability of the processes rather than in the implementation

[1]http://www.admire-project.eu

[2]This section is a summary of the DISPEL description found in the document *ADMIRE D 1.9 Final report on the ADMIRE model, language and ontology.* Document available at: http://www.admire-project.eu/docs/ADMIRE-D1.9-final-iteration.pdf

[3]see http://www-01.ibm.com/software/integration/wmqwf/

[4]http://www.oracle.com/technetwork/middleware/human-workflow/

details. The separation between the specification and the implementation is achieved by means of gateways that execute concrete workflows which implement the DISPEL specifications. A gateway may have numerous means to implement the same abstract workflow under different circumstances.

In DISPEL workflows are compositions of *processing elements* which receive, process, and produce streams of data. A processing element is defined in terms of the input and output stream of data. A stream of data is a continuous flow of data. In DISPEL input and output streams are called *connections*. A connection interface (see listing 4.1) is specified by the definition of i) the *structural type* of the pieces of data in the stream, ii) the *domain type* of the streamed data, and iii) an identifier.

$$Connection[: StructuralType][:: DomainType]\ identifier \tag{4.1}$$

Structural type refers to the data type, and it can be one of the basic types including *Integer*, *Real*, *String*, and *Boolean*, or an array, or a tuple. Arrays are lists of elements of a given type (*e.g.*, String[ ] addresses = new String[size]). On the other hand, tuples are unordered lists of elements which can have different types (*e.g.*, <Real lat, Real long, string name> GeoPosition = <40.418889,-3.691944,"Madrid">). Besides predefined structural types, users can create their own structural types by using the *Stype* keyword.

Domain types are used to make explicit the semantics of the streamed data. This semantics is defined by means of ontologies. For instance, a connection defining a stream of strings such as *Connection:String str* does not provide any formal description of the data stream semantics. Note that, even if the connection identifier is representative of the semantics of the data, this information cannot be leveraged automatically by a program since it conveys only symbolic information. Thus, to define the semantics of the data stream we use as a domain type an ontology class: *Connection:String::http://dbpedia.org/ontology/Country str*. In this case we state that the strings in the stream are instances of the class *Country* which is part of the DBpedia ontology[1]. Since the DBpedia ontology is written in OWL (Web Ontology Language) other programs can rely on it to elicit more information about the semantics of the stream, for instance, by inferring or reasoning over the ontology. In this chapter we

[1]The DBpedia ontology can be found at `http://wiki.dbpedia.org/Downloads37`

define the semantics of the input and output data of the different processes making up the method by means of the ontology presented in Annex A and the resource description framework RDF[1].

Besides the basic processing elements provided by the language, DISPEL supports complex processes where other processing elements can be orchestrated in an ordered sequence, therefore defining new workflows. Furthermore, the language allows users to define their own processing elements. To do so, users can create *processing element types* and then instantiate processes of these types. For complex processes defining a workflow users have to do four things: i) create a processing element type to identify the complex process, ii) define a *constructor function* to create a processing element of the defined type, iii) create a processing element of the defined type by calling the constructor function, and finally iv) create an instance of the processing element so that it can be executed or used as part of another workflow. A workflow is defined by instantiating the processing elements involved in the workflow, and then connecting their inputs and outputs so that the workflow is established.

To illustrate the workflows we use the graphical representations depicted in figure 4.2. The left hand side of the figure shows the icons used to represent the start and the end of the workflow, as well as the icons used to connect the inputs and outputs of the processing elements, and to explicitly redirect the workflow to a given processing element. The figure in the middle represents a processing element instantiation. In this figure the top and the bottom rectangles represent the input and output stream respectively. The rectangle in the middle contains the name of the processing element instance and its type. Finally, the figure in the right hand side represents a complex process created from a function. In this figure, besides the input, and output streams, and the name of the processing element instance, there is a rectangle where the name of the function and the type of the processing element returned by the function are specified. Below the function name there is a space to define the input arguments of the function.

Finally we want to note that the DISPEL code presented in this chapter is not executable since the programs implementing the details of each process in the method are not deployed yet in a gateway from which they can be executed.

[1]More information about RDF semantics available at `http://www.w3.org/TR/rdf-mt/`

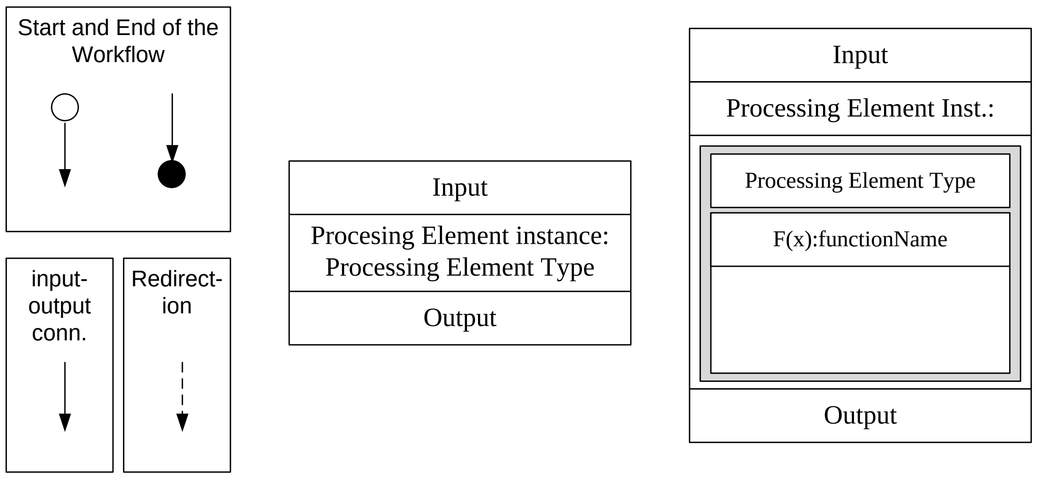

Start and End of the Workflow
input-output conn.
Redirect-ion
Input
Procesing Element instance: Processing Element Type
Output
Input
Processing Element Inst.:
Processing Element Type
F(x):functionName
Output

Figure 4.2: Graphical units to represent DISPEL workflows adapted from Martin and Yaikhom (2011).

4.1.4 Classification system model

The first challenge we have to overcome is to formalize in a unique manner the aforementioned user-generated classification systems: folksonomies and named-lists. This formalization is useful to show precisely, in a mathematical notation, how the data stream changes when it is transformed by each process in the method. In the lowest level of detail these systems consist of triples that relate a user, a category name, and a resource. We call these triples classification instances. Category names represent tags and names of lists. Thus, a user-generated classification system is a set of classification instances $CI \subseteq U \times CN \times R$ where U, CN and, R are finite sets whose members are users, category names and resources respectively. We define an individual classification instance in DISPEL by means of a custom structural type called *ClassificationInst* (see listing 4.1).

```
package es.upm.oeg{
  Stype ClassificationInst is
    <String user, categoryName, resource>;
}
```

Listing 4.1: Classification Instance definition in DISPEL.

This structural type is a tuple of three strings corresponding to a user, a category name, and a resource. Thus the whole classification system is represented as a finite stream of instances of type *ClassificationInst*. In addition, the listing shows that the description of the method is placed in the *es.upm.oeg* package. This package keeps in the same *namespace* all the components of the method, that is structural types, processing elements, and functions.

4.1.5 Illustrative Example

Throughout this chapter we use a small user-generated classification system to exemplify how each of the different components of the method transforms the inputs into the outputs to produce an ontology in the programming domain. This classification system, presented in table 4.1, is an extension of the folksonomy used in the state of the art section (see 2.3.3). The example shows annotations of distinct users over resources related to two distinct topics: coffee and programming. Though coffee and programming are totally different topics, some of these resources share user annotations. Note that the classification system has been represented as a list of classification instances where each row in the table is a triple that follows the definition given in listing 4.1.

Table 4.1: Example of classification instances representing collaborative tagging over some resources.

User	Category	Resource	User	Category	Resource
A	Coffee	R1	B	01seq†	R2
A	Java	R1	C	language	R2
A	Brew	R1	C	Programming†	R2
E	coffee†	R1	D	Languag†	R2
E	Organic†	R1	D	Program	R2
E	FairTrade†	R1	D	cs4500†	R2
B	Java	R1	D	Program	R3
B	Java	R2	D	Coding	R3
B	Language	R2	D	opensource†	R3

Triples marked with a † have been included or modified with respect to the original example presented in section 2.3.3.

In this example we have included different tag characteristics and annotation patterns that can be found in real life folksonomies. For instance, we have ambiguous tags (*java*), synonyms (*programming* and *coding*[1]), misspellings (*Languag*), composite keywords (*opensource*), nonsense words (*cs4500*), and mixed use of lower- and upper-case letters (*Language* vs *language*). With respect to annotation patterns, the example shows different users annotating the same resource (users A and E annotate resource R1), resources annotated by different users (Resource R2 is annotated by users B and C), resources sharing annotations (Resources R1 and R2 are annotated with the tag *Java*), and users sharing annotations (Users B and A use the tag *Java*).

In the rest of this chapter we describe our method which consists of two main processes: *Terminology Extraction* and *Semantic Elicitation*. The Terminology Extraction process is in charge of collecting, from the classification system, a set of terms relevant in the domain. The Semantic Elicitation process identifies the semantics of the terms gathered in the previous process. In figure 4.3 we show the processes and activities involved in the method, as well as the user roles participating in each activity. To describe each component (*i.e.*, process, activity, or task) we use the following template. First we present the component objective followed by guidelines for Ontology Engineers and Domain Experts about how to carry out the given component. Next we present the formalization of the component using DISPEL, and finalize the description exemplifying its input and output using the illustrative example data. Note that guidelines and examples are sometimes postponed to more detailed components or to the sections where the workflows are presented.

4.2 Process 1. Terminology extraction

The objective of the Terminology Extraction process is to gather from the user-generated classification system a set of terms that are relevant to a given domain. By relevant we mean terms that are commonly used in a particular domain. Commonly used terms can be identified by examining terminology resources used in the domain including glossaries and thesauri (e.g., Yahoo! financial glossary[2] or Yahoo! education thesaurus[3]). This process queries the classification system and receives a stream of raw data. Thus it must face relevant and non-relevant and possibly noisy data. Once we have gotten rid of noisy

[1]Programming and coding are synonyms in the sense of *Computer programming.*
[2]http://biz.yahoo.com/f/g/
[3]http://education.yahoo.com/reference/thesaurus/

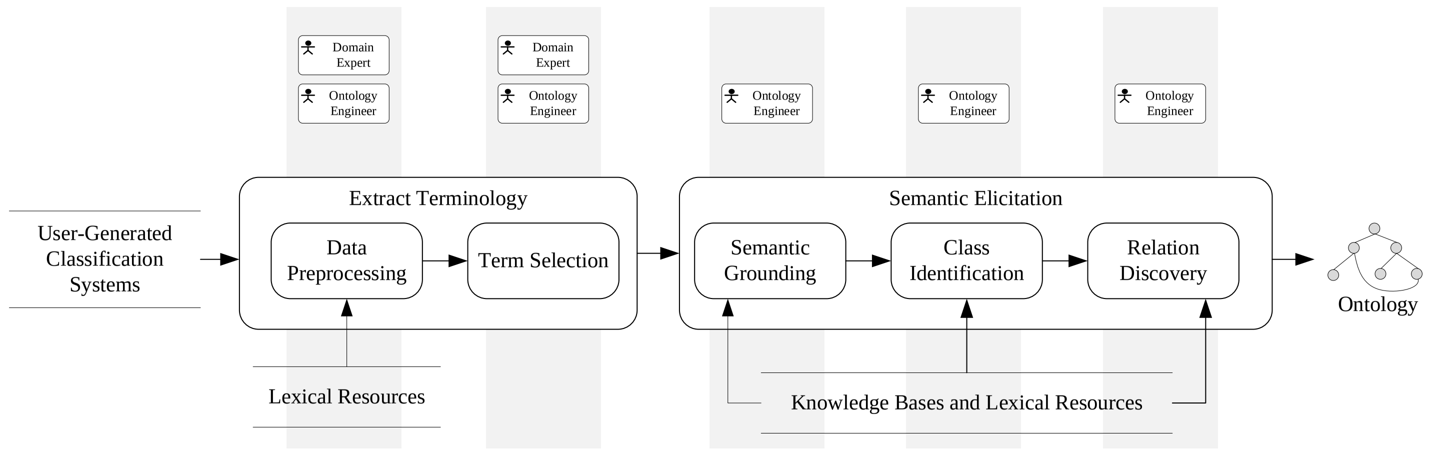


Figure 4.3: Processes of the method for developing ontologies from user-generated classification systems.

data we have to represent the data according to a predefined model that we can exploit later for the collection of the domain relevant terms. Consequently, to obtain data from the user-generated classification system and filter out noisy data we propose a Data Preprocessing activity. Then we propose to carry out a Term Selection activity which processes the resulting data from the Data Preprocessing activity to extract the set of domain relevant terms. During this process, ontology engineers and domain experts collaborate to define the classification system to be tapped, the criteria used to retrieve data from the data source, and the strategy to gather the set of relevant terms.

Formalization. In listing 4.2 we present a preliminary formalization of the Extract Terminology process which is going to be refined later through the description of its activities and tasks. The process is defined by means of a processing element called *ExtractTerminology*. This type defines a processing element that does not receive any data and generates a stream of contextualized terms. The lack of an input data stream to the processing element shows that the access to the classification system data is managed by an internal activity. The output of the processing element is a stream of tuples defined by the structural type *ContextualizedTerm*. Each tuple consists of a given term, and its context defined as an array of classification instances from which the term was obtained. The semantics of each tuple of *ContextualizedTerm* is defined by the domain type *ugcs:ContextualizedTerm*. *ugcs:ContextualizedTerm* is a concept in the ontology presented in annex A. A processing element of type *ExtractTerminology* is created by means of the function *extractTerminologyFn* which receives two input arguments. The first argument *UGCS* specifies the user-generated classification system from which the data are going to be extracted, and the second argument *lexicalResource* indicates the lexical resource that may be used in the normalization of category names.

4.2.1 Activity 1.1. Data preprocessing

The objective of this activity is to obtain data from the classification system, which are then cleaned, normalized, and transformed into a specific representation useful for the subsequent activities. Data Preprocessing is a regular activity in any knowledge discovery process since it guarantees the quality of the data over which the process is going to be applied and therefore this activity influences the results of the whole process (Pyle, 1999). In our method we include tasks such as *Extraction*, *Normalization* and *Transformation*. The Extraction and Normalization activities filter out non-relevant

```
package es.upm.oeg{
  namespace ugcs "http://www.upm.es/ontologies/ugcs#";

  Stype ContextualizedTerm is
    <String term,[ClassificationInst] context>;

  Type ExtractTerminology is
    PE(<> =>
       <Connection:ContextualizedTerm::"ugcs:ContextualizedTerm" term>);

  //Constructor of processing elements of type ExtractTerminology
  PE(ExtractTerminology) extractTerminologyFn(String UGCS, String lexicalResource){
    ...
  }
}
```

Listing 4.2: Definition of the processing element type representing partially the Terminology Extraction process

data while extracting and standardizing the input data respectively. The Transformation task maps the data from their current format to the format expected by the Term Selection activity.

Formalization. In listing 4.3 we present the preliminary formalization of the processing element type *PreprocessData* which represents the Data Preprocessing activity. This type does not receive any input data since the access to the data is delegated to the Extraction task. On the other hand, the output is a stream of data which is not constrained to any DISPEL or user generated structural type. This lack of a structural type represented by the keyword *Any* in the output connection is intentional since the data representation to which the data is transformed during this activity depends on the techniques that are going to be applied in the Term Selection activity. A processing element of type *PreprocessData* is created by means of the function *preProcessDataFn*. This function receives as arguments the user-generated classification system UGCS from which the data are going to be extracted, and the lexical resource used in the normalization task.

```
package es.upm.oeg{
  Type PreprocessData is
    PE(<> =>
       <Connection:Any transformedInst>);

  //Constructor of a processing element of type PreprocessData
  PE(PreprocessData) preProcessDataFn(String UGCS, String lexicalResource){
     ...
  }
 }
```

Listing 4.3: Partial definition of the processing element type representing the Data Preprocessing activity

4.2.1.1 Task 1. Extraction

The Extraction task obtains from the source the initial set of classification instances. The objective is to make data available to the rest of activities in a single format independent from the data source proprietary representation. This task input is defined by data about the resource to be accessed including its location. The output is a set of classification instances which may contain data that is relevant to the domain of study.

Guidelines. This activity should be accomplished with the help of domain experts and ontology engineers. First they have to explore the candidate user-generated classification systems **to analyze whether a particular system contains data (i.e., category names) related to the domain of study**. The objective is to select a classification system to be used as data source during the rest of the process.

Besides relevant category names, ontology engineers have to take into account other criteria such as data availability, and publication constraints and mechanisms. **The ontology engineer is responsible for addressing the technical aspects of the data collection and assessing its feasibility.** Normally web applications allow accessing their data but they also define publication and query constraints to avoid system overload. For instance, some social networks provide web services to query their data but they impose usage constraints such as a maximum number of requests per day or a minimum delay between requests. In these cases the information extraction have to be incremental. That is an initial set of classification instances is retrieved according to some query criteria. Then these instances are used to query for instances sharing similar characteristics.

In this activity it is possible to implement filters allowed by the data publication mechanism. These filters were identified from the review of the state of the art and from the expertise gained when accessing these data sources in our experiments. They are classified in structural, group, and lexical filters:

- **Structural filters** are used to select data according to the components of the classification systems and their data properties. Thus we can specify a structural filter to return classification instances according to a user, a category name, a resource or the combination of any of them. Moreover, we can narrow the search to retrieve classification instances according to properties such as creation date or location in case this information is available.
- **Group filters** allow selecting data according to the value of functions that can be applied on groups of classification instances. In our case, we consider valid functions those specified in the SQL group functions (*i.e.*, count, sum, max, min, avg, distinct). For instance, in the case of collaborative classification systems where some labels are used to categorize resources by one or more users we can filter data based on usage statistics of the labels (*e.g.*, frequency of annotation) to gather the vocabulary agreed by users around resources. In addition, we can use these filters to retrieve only instances where the resource has been classified under a minimum number of categories.
- **Lexical filters** are applied on the string of characters associated with each piece of data, and hence they allow selecting and discarding data according to lexical characteristics such as the string length, or the type of characters found in each string. These filters are suitable to discard, from the beginning, category names that can be considered noise such as those mixing numbers and characters, or non alphabetical characters.

In general we recommend to extract the data following standard practices such as the ones defined in the *Extract, Transform and Load* (ETL) processes (Kimball and Caserta, 2004). In our experiments presented in chapter 6, we extracted data from *comma-separated values* (CSV) dumps provided by Delicious (see section 6.2) and from the web services providing access to Twitter data (see section 6.4). In the first case we loaded the data in a relational database and used SQL queries to extract the portion of data we were interested in. In the second case we used the web services provided by Twitter to get the data and we store them in a relational database.

Formalization. To represent the extraction task in DISPEL we define, in listing 4.4, the processing element type *Extract*. A process element of this type is in charge of retrieving the classification instances from the user generated classification system. A processing element of type *Extract* is created by means of the function *extractFn*. This function receives as argument the user-generated classification system UGCS from which the data are extracted. The actual implementation of the data extraction is the function *es.upm.oeg.DataExtractionImpl*.

```
package es.upm.oeg{
  //Implementation of a process to extract data from a UGCS
  use es.upm.oeg.DataExtractionImpl;

  //Processing element type for the extraction process
  Type Extract is
    PE (<> => <Connection:ClassificationInst::"ugcs:ClassificationInst" inst>);

  //Function to create a processing element of type Extract
  PE(Extract) extractFn(String UGCS){
    //Instantiation of the program to extract data
    DataExtractionImpl dataExtraction = new DataExtractionImpl;

    //Setting the source of data
    |-UGCS-|=>dataExtraction.inputDataSource;

    //return the classification instances from the UGCS
    return(<>=>
           <Connection inst=dataExtraction.ouputClassificationInst>);
  }
}
```

Listing 4.4: Definition of Extraction task in DISPEL

Example. To exemplify the extraction task we describe the process followed to obtain the classification instances which make up our example data (table 4.1). We, as ontology engineers, decided to implement an incremental retrieval of data. First we query the data source for resources annotated with the tag *programming*. This query returns the resource R2. Then we gathered all the classification instances where R2 appears, and query the data source for resources annotated with any of the tags used to annotate R2, which include tags such as *java*, *language*, and *program*. These queries retrieve resources R1 and R3 from which we collect their classification instances. Starting from here the process becomes recursive. We use the set of tags used to annotate the retrieved resources for querying more resources which in turn are used to collect their tags so that another query loop can start. The output of the extraction task are the classification instances presented in table 4.1 which are arranged in a stream of data according to the structural type *ClassificationInst*: *<A,Coffee,R1>,<A,Java,R1>,...,<D,Coding,R3>,<D,opensource,R3>*.

4.2.1.2 Task 2. Normalization

The Normalization task aims at standardizing category names in classification instances by converting them to a normative reference. In the case of user-generated classification systems this normalization is important since category names usually contain spelling variations that may lead to consider two pieces of data as distinct when they actually are referring to the same thing. These spelling variations can be due to misspellings, to the use of plurals, to the mix of upper and lower cases within words, to the different characters used to separate two words composing a name, or to the use of concatenated words. The output of this activity is a stream of normalized classification instances where category names generated by users have been normalized by following a standard representation.

Guidelines. Ontology Engineers and Domain Experts must **agree on a standard way for representing category names**. Examples of standard representations are words in upper or lower case, words starting with a capital letter and the rest in lowercase, etc. Another option is to adhere to a third party standard defined, for instance, by a dictionary or by an encyclopedia. For this task, ontology engineers can use *stemmers* to reduce words to their root form which can be thought as form of normalization.

We present, in section 5.1.1 a technique to normalize category names based on approximate matching of the category names to dictionary entries. In addition we have carried out an experiment with category names extracted from Delicious in section 6.1.

Formalization. Listing 4.5 presents the DISPEL representation of the normalization task. We have defined a structural type *NormalizedClassificationInst* which adds to the classification instance a string representing the normalized version of the category name. Having specified what a normalized classification instance is, we define a processing element type *Normalize* where the input is the stream of classification instances retrieved in the extraction task, and the output is the stream of normalized classification instances. A processing element of type *Normalize* is created by means of the function *normalizeFn* which receives as argument the lexical resource to be used during the normalization. The actual implementation of the normalization program is the function *es.upm.oeg.NormalizationImpl*.

```
package es.upm.oeg{
  //Implementation of a program to normalize category names
  es.upm.oeg.NormalizationImpl;

  //Structural type defining a normalized classification instance
  Stype NormalizedClasInst is
    <String user, categoryName, normalizedCateName, resource>;

  //Processing element type of a Normalize process
  Type Normalize is
  PE(<Connection:ClassificationInst::"ugcs:ClassificationInstance" inst>
     =>
     <Connection:NormalizedClasInst::"ugcs:NormalizedClasInst" normalizedInst>
    );

  //Function to create a processing element of type Normalize
  PE(Normalize) normalizeFn(String lexicalResource){
    //Instantiation of the program to normalize category names
    NormalizationImpl normalizationImpl = new NormalizationImpl;

    //Setting the lexical resource to be used in the normalization
    |-lexicalResource-|=>normalizationImpl.lexicalResource;

    //Connect the input and output streams of the processing element to
    //the input and output streams of the program
    return(<Connection inst=normalizationImpl.inputInst>
           => <Connection normalizedInst = normalizationImpl.outputNormalizedInst>);
  }
}
```

Listing 4.5: Definition of the Normalization task in DISPEL

Example. To normalize the category names found in the stream of classification instances we use the WordNet lexicon (Fellbaum, 2005). First, we query WordNet with each category name. In case we find an entry we adhere to the WordNet lexical representation of this entry. We found entries for all the category names less *FairTrade, 01seq, Languag, cs4500*, and *Opensource*. For these cases we use an spelling service (*e.g.*, the Yahoo! spelling service). These services splits concatenated words, and produces suggestions for misspelled words. While we did not get any suggestion for *01seq* and *cs4500*, we got the following suggestion for the rest: *fair trade*, *Language* and *open source*. We search for these suggestions in WordMet and found entries for all of them. Therefore the output of this task is the stream of normalized instances which are presented in table 4.2.

Table 4.2: Normalized classification instances following WordNet entries

<A,Coffee,coffee,R1>	<B,Language,language,R2>
<A,Java,Java,R1>	<C,language,language,R2>
<A,Brew,brew,R1>	<C,Programming, programming,R2>
<E,coffee,coffee,R1>	<D,Languag,language,R2>
<E,Organic, organic,R1>	<D,Program,program,R2>
<E,FairTrade,fair trade,R1>	<D,Program,program,R3>
<B,Java,Java,R1>	<D,Coding,coding,R3>
<B,Java,Java,R2>	<D,opensource,open-source,R3>

Note that spelling suggestion services can induce some errors since these services provide suggestions according to lexical similarities of words to concept names. For instance, a misspelled category name such as *tre* may be presented with suggestions such as *trek*, *tree* and *trevon*. All of them are valid suggestions but a normalization process will pick up one of them. Thus, when designing the normalization strategy of category names we have to bear in mind the trade-off between normalizing a large number of category names and the accuracy of the normalization.

4.2.1.3 Task 3. Transformation

The final task in the Data Preprocessing activity is the Transformation where the normalized data are converted to a particular schema adapted to the needs of the Term

Selection activity which is the next process in the workflow. In this activity it is possible to map data from a lower level of granularity to a higher level.

Guidelines. The Ontology Engineer must select the target schema according to the information view required during the Term selection, which is the activity where the data is going to be processed. However **different schema representations affect the computational complexities of the operations performed over the data**. For instance, classification systems can be represented as hypergraphs where nodes are the different components of the classification systems and the edges between nodes represent relations between the components. Hypergraphs can be modeled in different ways such as adjacency lists, adjacency matrices, incidence lists, and incidence matrices (Cormen et al., 2001). Each of these representations uses a data structure to store the graph in different ways. This data structure affects the cost of maintaining operations and of the traversing of the graph. Adjacency lists are generally preferred because they efficiently represent sparse graphs while adjacency matrices are preferred if the graph is dense (*i.e.*, the number of edges is close to the number of vertices squared) (Cormen et al., 2001).

Moreover **hardware constraints regarding the memory size, or CPU speed can affect the selection of a representation schema**. For instance the dimension of a matrix used to represent a classification system containing a large amount of resources and categories may not fit in memory due to the high number of cells in the matrix. A valid alternative is to keep the information in a relational database from which the data can be accessed efficiently.

The Transformation task and the Term Selection activity have a high coupling since the needs of the latter influences the output of the former. Thus **according to the algorithm to be used in the Term Selection activity we may need to obtain a different view of the classification instances** which can show the data with a distinct level of granularity. A possibility is to create a view using the relational operator of projection and the aggregation function (Date, 2005). Projection allows selecting the subset of the classification instances that we are interested in, while aggregation enables to calculate statistics over defined groups of instances. Let us suppose we are interested in a view of the classification system describing the category names used over resources along with the number of users that have placed each resource in a specific category. This view can be created first by grouping classification instances by category names, and resources, counting the number of elements in each group, and finally projecting the category name and the resource identifying each group along with the element count.

We recommend to follow the standard transformation practice defined in the *Extract, Transform and Load* (ETL) processes (Kimball and Caserta, 2004). The transformation step of this process is entirely carried out in a relation database environment and therefore data can be transformed using all the capabilities of the SQL language.

In section 5.1.2 we present a technique to carry out the Term Selection activity based on a graph where the vertices are resources in the user-generated classification system and there exists an edge between two resources if they share annotations. Therefore the original user-generated classification system have to be transformed to this graph. In section 6.3 we present an experiment where we apply this technique over Delicious data. The data to be transformed (*i.e.*, the classification system) wer stored on a relational table. We transformed these data, using SQL queries and intermediate tables, to the graph required by the Term Selection activity which was represented as an adjacency list stored in a relational table.

Formalization. To represent this task in DISPEL we define a processing element type *Transform* (see listing 4.6). This processing element receives the normalized classification instances and returns tuples representing the new view of the data. The *<rest>* specification in the output of the processing element type means that we do not constrain the structural type of the tuples returned by this task, though we require tuples instead of any other type. A processing element of type *Transform* is created by means of the function *transformFn*. This function creates an instance of the program *TransformImpl* which is the actual implementation of the transformation process.

Example. In this task we group the classification instances according to shared annotations over resources. That is we group them by normalized category names and resources, and add the count of users who have agreed with this annotation. Therefore we transform the normalized instances into triples of the form: *<normalizedCategoryName, resource, annotationFreq>*. Thus the output of this task is the stream of transformed instances presented in table 4.3.

4.2.1.4 Choreography of the data preprocessing activity

To put together the tasks of the Data Preprocessing activity we define a workflow that connects them in an ordered sequence. This workflow definition is presented in listing

```
package es.upm.oeg{
  //Implementation of a program to transform normalized classification instances
  use es.upm.oeg.TransformImpl;

  //Processing Element Type of a transformation process
  Type Transform is
    PE (<Connection:NormalizedClasInst::"ugcs:NormalizedClasInst" normalizedInst>
        =>  <Connection:<rest> transformedInst>);

  //Function to create a processing element of type Transform
  PE(Transform) transformFn(){
    //Instance of the program to transform the data
    TransformImpl transformImpl = new TransformImpl;

    //Connecting the input of the processing element to the input of the program
    //and the output of the program to the output stream of the processing element
    return (<Connection normalizedInst=transformImpl.inputNormalizedInst>
            => <Connection transformedInst=tranformImpl.outputTransformedData>)
  }

}
```

Listing 4.6: Definition of the Transformation task in DISPEL

4.7 and depicted in figure 4.4. First we define a function *preProcessDataFn* which creates a processing element of type *PreprocessData* (see listing 4.3). This function creates processes for the Extraction, Normalization and Transformation tasks which are then orchestrated as follows. The Extraction process output, which is a stream of classification instances, is connected to the Normalization process input. The Normalization process output (*i.e.*, the normalized classification instances) are connected to the input of the Transformation process. Finally the transformed classification instances which

Table 4.3: Classification instances grouped according to category names and resources.

<coffee,R1,2>	<fair trade,R1,1>	<programming,R2,1>
<Java,R1,2>	<Java,R2,1>	<program,R3,1>
<brew,R1,1>	<language,R2,3>	<coding,R3,1>
<organic,R1,1>	<program,R2,2>	<open-source,R3,1>

are the output of the Transformation process, are sent through the output of the Data Preprocessing activity.

```
package es.upm.oeg{
  PE(PreprocessData) preProcessDataFn(String UGCS, String lexicalResource){

    //Defines the processing elements for each task
    PE<Extract> Extraction = new extractFn(UGCS);
    PE<Normalize> Normalization = new normalizeFn(lexicalResource);
    PE<Transform> Transformation = new transformFn();

    //Instances of the processes
    Extraction extractProcInst = new Extraction;
    Normalization normalizeProcInst= new Normalization;
    Transformation transformProcInst = new Transformation;

    //Workflow definition (connecting inputs and outputs)
    extractProcInst.inst => normalizeProcInst.inst;
    normalizeProcInst.normalizedInst => transformProcInst.normalizedInst;

    //Delivers the transformed instances through the output of the preprocessing process.
    return PE(<>
              => <Connection transformedInst=transformProcInst.transformedInst>);
 }
}
```

Listing 4.7: Definition of the Data Preprocessing workflow in DISPEL

4.2.2 Activity 1.2. Term selection

The goal of the Term Selection activity is to gather, from the set of classification instances delivered by the Data Preprocessing activity, a set of terms that are relevant in a particular Domain. Therefore the input of this activity are the classification instances obtained by the Preprocessing activity and the output is a list of terms related to the domain of study. Besides the terms themselves it is important to include contextual information that help to understand in subsequent steps, their meaning. This context can include provenance information so that we can reconstruct the relations of a term with the components of the classification systems. In other words, given a term we should be able to obtain all the information about users, resources and category names

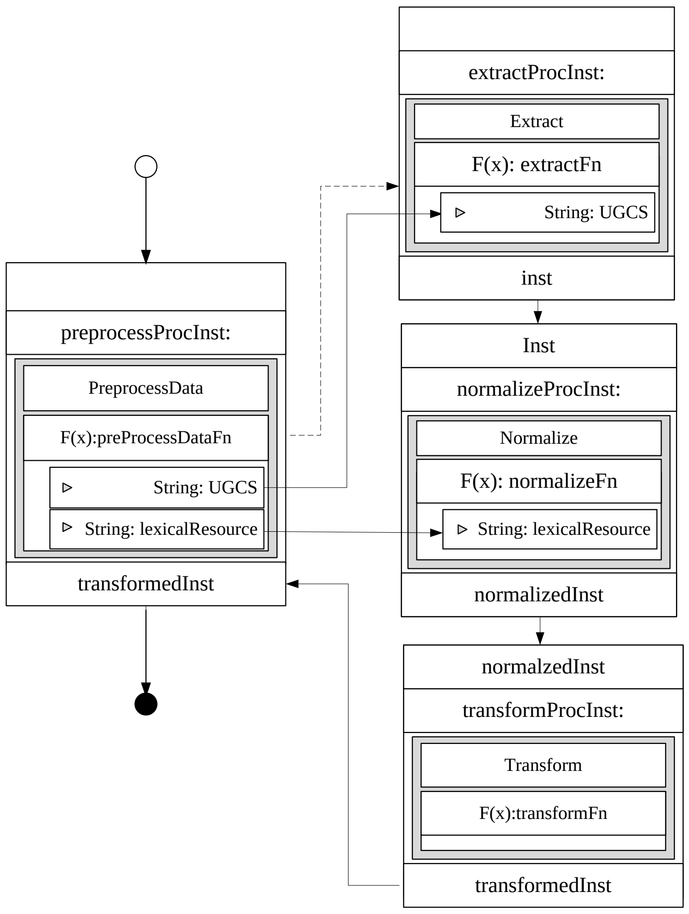


Figure 4.4: Graphical representation of the data preprocessing activity

that are related to that term.

Guidelines. During this activity Domain Experts and Ontology Engineers work together to define the strategy to collect the relevant terms from the category names of the classification system. In general, in this process the whole data set has to be traversed in order to analyze each piece of data regarding the domain of study. Thus some **heuristics have to be defined to collect the relevant domain terms**.

For instance, an heuristic can be based on the assumption that *relevant domain resources are annotated with relevant domain terms.* Thus if we identify domain relevant resources in the classification system we can collect the most frequent category names under which they are classified as relevant domain terms. Given the amount of information in user-generated classification systems *the identification of domain rele-*

vant resources should be automatic. For instance by comparing the resources with those listed in a source of relevant domain resources such as an Internet directory. Another possibility is to use an initial subset of relevant domain resources, automatically or manually defined, and then find similar resources in the user generated classification system. This last option requires the definition of a similarity measure between resources.

In case the set of instances is very large, traversing it can be a time and resource consuming operation. Thus some **strategies can be defined to limit the traversing of the classification instances to some of them that fulfill certain conditions**. A possibility to reduce the dataset is to use statistical sampling since it allows extracting a representative set of data according to predefined error rate. Nevertheless, when the input data is a continuous stream of data it is not possible to use traditional sampling and therefore more appropriate sampling techniques for dynamic data streams have to be used.

In section 5.1.2 we propose a technique to carry out this activity which is based on a graph representation of the user-generated classification systems where the vertices are resources and there exists an edge between two resources if they share annotations. We use spreading activation (Crestani, 1997) to traverse the graph and collect domain relevant terms. In section 6.3 we present an experiment over Delicious data where we show how applying this technique we are able to obtain a domain terminology.

Formalization. This activity is formalized in listing 4.8 through a *SelectTerms* processing element type. The input is a stream of transformed instances which are not tied to any structural type while the output is a stream of contextualized terms. A contextualized term was defined in listing 4.2 as a structural type that represents a tuple containing the term itself along with a list of classification instances from which the term was obtained. A processing element of type *SelectTerms* is created by means of the function *selectTermFn*. This function creates an instance of the program *Select-TermsImpl* which is the actual implementation of the term selection process.

Example. To exemplify the extraction of terms we use a heuristic based on the relevance of the annotated resource to the programming domain. We use a web directory of programming-related websites where we look for the resources in our classification systems. If a resource exists in the directory then we conclude that it is relevant for the domain and use its category names as relevant terms. For this example we use the

```
package es.upm.oeg{
  //Implementation of a program to select terms
  es.upm.oeg.SelectTermsImpl;

  //Processing element type for the selection of terms
  Type SelectTerms is
    PE ( <Connection:Any transformedInst>
         => <Connection:ContextualizedTerm::"ugcs:ContextualizedTerms" contextualizedTerm>);

  //Function to create a processing element of type SelectTerms
  PE(SelectTerms) selectTermFn(){
    //Instance of the program to select terms
    SelectTermsImpl selectTermsImpl = new SelectTermsImpl();

    //Connects the input defined in the processing element type to the input of the program
    //and the output of the program to the output of the processing element type
    return (<Connection transformedInst = selectTermsImpl.inputData>
            => <Connection contextualizedTerm = selectTermsImpl.selectedTerms>);
  }

}
```

Listing 4.8: Term Selection activity in DISPEL

programming category in the Yahoo! directory[1]. Thus from the resources in the stream of transformed instances presented in 4.3 we found in this directory the resources R2 and R3. Therefore we get rid of R1 and produce a stream of terms with the category names under which R2 and R3 are annotated (see table 4.4).

Table 4.4: Tuples of relevant terms extracted from the transformed instances

<Java,[<Java,R2,1>]>	<program,[<program,R2,2>,<program,R3,1>]>
<language,[<language,R2,3>]>	<programming,[<programming,R2,1>]>
<coding,[<coding,R3,1>]>	<open-source,[<open-source,R3,1>]>

[1] http://dir.yahoo.com/Computers_and_Internet/Programming_and_Development/

4.2.3 Choreography of the terminology extraction process

So far we have defined the Data Preprocessing and Term Selection activities that are carried out during the Extract Terminology Process. To assemble these activities within the process we create the workflow described in listing 4.9 and depicted in figure 4.5. The Extract Terminology process is created by the function *extractTerminology*. First, a Data Preprocessing process and a Term Selection processes are created. Next the output of the Data Preprocessing activity, which is the set of transformed classification instances obtained from the classification systems, is connected to the input of the Term Selection activity. Finally the set of contextualized terms produced by the Term Selection is delivered through the output of the Terminology Extraction process that is returned by the function *extractTerminologyFn*.

```
package es.upm.oeg{
  PE(ExtractTerminology) extractTerminologyFn(String UGCS, String lexicalResource)  {

  //Defines the processing elements of the activities
  PE<PreprocessData> DataPreprocessing = new preProcessDataFn(UGCS, lexicalResource);
  PE<SelectTerms> TermSelection= new selectTermsFn();

  //Instantiates the processes to be used during this workflow
  DataPreprocessing preprocessProcInst = new DataPreprocessing;
  TermSelection selectTermsProcInst= new TermSelection;

  //Defines the workflow (connecting inputs and outputs)
  preprocessProcInst.transformedInst => selectTermsProcInst.transformedInst;

  //Delivers as output the stream of contextextualized terms
  return PE(<> =>
            <Connection contextualizedTerms = selectTermsProcInst.contextualizedTerms>);
 }
}
```

Listing 4.9: Workflow for the Extract Terminology process

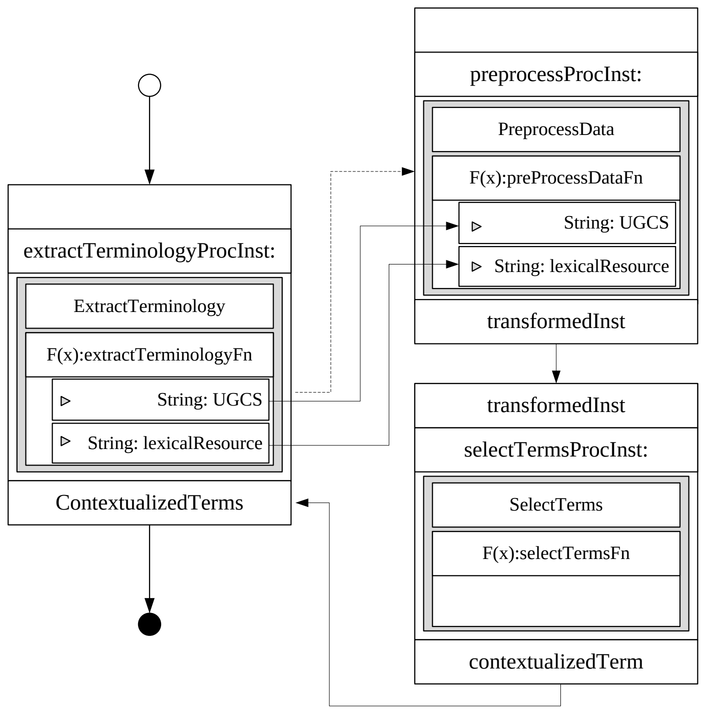


Figure 4.5: Graphical representation of the extract terminology process

4.3 Process 2. Semantic elicitation

The goal of the semantic elicitation process is to obtain a domain ontology from the relevant domain terms found in the Extract Terminology process. To achieve this objective we propose to reuse classes and relations from existing knowledge bases which are then used to create the ontology. To do so we propose the following activities: i) Semantic Grounding where terms are linked to entities in the knowledge base, ii) Identification of Classes where these entities are used to look for classes in the knowledge base that represent their meaning, and iii) Discovery of Relations where we look for existing relations between the classes in the knowledge base. With the set of classes and relations found during the process we produce the initial domain ontology.

The success of this process depends, to a large extent, of the knowledge base used. The ontology engineer is responsible for the selection of the knowledge base from which the semantic entities are obtained. He must take into account the **coverage of the knowledge base with respect to the domain of study**. A poor domain coverage may lead to produce a small ontology with rather general concepts. To improve coverage more than one knowledge base can be used, though this solution can introduce new problems arising from **heterogeneous representations** and from **semantic mismatches** regarding the entities.

Heterogeneous representations refer to the fact that different knowledge bases can use distinct modeling languages, which in turn can be based on different representation paradigms. This heterogeneity affects the interfaces provided to query the knowledge bases which also can be potentially different.

On the other hand, semantic mismatches can emerge from the different conceptualizations of the same pieces of knowledge. Conceptualizations can vary according to the different point of view of creators, or to the requirements defined for the knowledge base, among other variables. Nevertheless, these issues can be addressed with the help of alignment and integration techniques.

In addition, ontology engineers may need to take into account the **lexical capabilities of the knowledge base** including the representation of possible concepts for ambiguous words, the representations of concepts that can be referred to with different synonyms, the availability of textual descriptions defining the concepts, and the representation of the different verb conjugations. Another aspect that may influence the selection of the knowledge base is **the multilinguality support**. Given that the source of the input terms is a set of category names written by users, they can be written in different languages according to the languages that users use to write. Therefore, if ontology engineers decide to tap into the potential multilingual terms, they have to select a knowledge base with multilingual information.

To avoid the aforementioned problems we recommend to use knowledge bases: i) which are comprehensive so that domain knowledge is well covered, ii) which are published in standard languages such as RDF, RDFS or OWL, iii) for which the quality of the links to other data sources have been assessed, and iv) which contain information of different meanings of words and the different words used to refer to those meanings.

We can find some knowledge bases in the **Web of Data** (Bizer et al., 2009a) which fulfill these characteristics. For instance DBpedia (Bizer et al., 2009b) and OpenCyc[1] are comprehensive and contains information of different domains. They are published in RDF and are available through sparql endpoints. The links between concepts of each of the knowledge bases are created based on the wikipedia pages which describe

[1]see http://www.opencyc.org/

each concept. In both cases the association between a concept and a wikipedia page is supervised by users. Finally, DBpedia has extracted information of Wikipedia disambiguation pages where candidate concepts for ambiguous words are specified, and of Wikipedia redirection pages where alternative words to refer to a given concept are defined. Therefore the techniques that we proposed in the next chapter (section 5.2) to carry out the Semantic Elicitation rely on knowledge bases published in the web of data.

Formalization. Before going into details of the process we need to define its input and output. The input is the stream of contextualized terms according to the structural type *ContextualizedTerm* which was described in listing 4.2. The output is an ontology that can be represented as a stream of statements. Each statement is a triple which consists of a subject, a predicate, and an object and its semantics is defined by the *rdf:Statement* class[1]. These statements allow defining the classes, relations and restrictions making up the ontology according to the syntax and semantics of an ontology language such as RDFS or OWL[2]. Thus, having specified the input and output we define the processing element type *ElicitSemantics* to represent this process (see listing 4.10). A processing element of type *ElicitSemantics* is created by means of the function *elicitSemanticFn*. This function receives as an argument the knowledge base from which we are going to reuse the conceptualizations.
In the following we present the activities involved in the Semantic Elicitation process, and finalize the section with the definition of the process workflow where the activities are arranged in a given order.

4.3.1 Activity 2.1. Semantic grounding

By semantic grounding we mean to give terms explicit semantics on the basis of their association with entities in knowledge bases. Therefore, the objective of this activity is to identify for each of the domain relevant terms, obtained in the Terminology Extraction process, the semantic entity that represents its meaning. A semantic entity is any component of the knowledge base that can be identified uniquely, for instance, by a URI. Hence, semantic entities are data and object properties, instances and classes.

[1]see http://www.w3.org/TR/rdf-schema/
[2]see http://www.w3.org/TR/owl-ref/

```
package es.upm.oeg{
  Stype Statement is
    <String subject, predicate, object>;

  Type ElicitSemantics is
    PE(<Connection:ContextualizedTerms::"ugcs:ContextualizedTerms" contextualizedTerm>
       => <Connection:Statement::"rdf:statement" ontologicalStatement>);

  //Constructor of a processing element of type ElicitSemantic
  PE(ElicitSemantics) elicitSemanticsFn(String knowledgeBase){
     ...
  }
}
```

Listing 4.10: Partial definition of the Processing Element Type for the Semantic Elicitation process

Guidelines. When ontology engineers are defining the strategy to select the concepts defining the meaning for each term they must face the ambiguity of some of them. Therefore to cope with **ambiguity** the strategy to be designed has to include a disambiguation approach. **Synonymy** also poses challenges since the approach must be able to identify when two terms refer to the same concept, so that this kinds of terms are correctly grounded.

In addition, **multilinguality** of terms (Gracia et al., 2012) may be considered since ontology engineers might have to design a procedure to identify the language of terms so that they can be grounded to the right entities. Similarly, ontology engineers have to deal with the different representations of multilingual concepts in the knowledge base. Two terms written in different languages can be represented by a unique concept or by two different concepts, one for each language. The former representation allows keeping track of cross-lingual synonyms, though it does not provide support for concepts which exist in one of the languages but not in the other, while the latter representation does it.

Distinct **strategies can be applied to select the right semantic entity for a term**. Among them we have **the selection of the most probable sense for a word defined by a group of users or according to statistics of usage of each meaning in a corpus or in the knowledge base**. Other strategies can use **the context of**

the word to select the right sense. These word sense disambiguation strategies has been tested in (Mendes et al., 2011). Finally the strategy can be automatic or assisted. An automatic strategy produces as result the most probable concept for each term. In contrast an assisted one provides a list of candidate concepts from which users can pick the right one (Gracia et al., 2010).

Given the amount of information presented in user-generated classification systems an assisted approach for the semantic grounding is unfeasible. In addition the approaches that give preference to the most frequent use of a word (*i.e.*, default sense or prior probability) may not select less frequent meanings which can be related to specific domains. Therefore we **recommend to use a sense disambiguation approach** that analyse the candidate meanings according to a context and select the one that better define the meaning of ambiguous word.

We propose a technique to carry out this activity in section 5.2.1 where we use the Vector Space Model (Salton and Mcgill, 1986) to select among candidates the concept that better defines the meaning of the category name. In section 6.2 we present the results of an experiment where we grounded a set of multilingual Flickr tags to semantic entities in DBpedia. In addition in section 6.3 we use this technique, within the framework of a process to develop an initial ontology in the stock market domain, to ground Delicious tags to semantic entities in DBpedia.

Formalization. Listing 4.11 presents the formalization in DISPEL of the semantic grounding activity. We define the processing element type *Ground*. The input is the list of contextualized terms which are passed directly from the input of the Semantic Elicitation process. The output is the list of unique semantic entities created from the union of the entities to which the terms were grounded. Each entity is represented by a string and its semantics is defined by the *rdf:Resource* class[1]. A processing element of type *Ground* is created by means of the function *groundFn*. This function receives as argument the knowledge base used for the semantic grounding of terms, and creates an instance of the process *GroundingImpl* which is the actual implementation of the semantic grounding process.

Example. To ground the stream of terms gathered from the classification system we use WordNet synsets. In WordNet (Fellbaum, 2005) every word is associated to one or more synsets, and each synset represents a possible meaning. Thus we retrieved

[1]see http://www.w3.org/TR/rdf-schema/

```
package es.upm.oeg{
  //Implementation of a program to ground terms to semantic entities
  use es.upm.oeg.GroundingImpl;

  //Processing element type defining a semantic grounding process
  Type Ground is
    PE(<Connection:ContextualizedTerms::"ugcs:ContextualizedTerms" contextualizedTerm>
       => <Connection:String::"rdf:Resource" entity>);

  //Constructor of the processing element of type Ground
  PE(Ground) groundFn(String knowledgeBase){
       //Instance of the program to ground terms to semantic entities
       GroundingImpl groundImpl = new GroundingImpl();

       //Set the knowledge base used for grounding the terms
       |-knowledgeBase-|=>groundImpl.knowledgeBase;

       //Connect the stream of contextualized terms with the input data of the program
       //and the output data of the program with the output stream of the PE type.
       return (<Connection contextualizedTerm = groundImpl.dataToGround> =>
               <Connection entity = groundImpl.entities>);
  }
}
```

Listing 4.11: Definition of the Processing Element Type for the Semantic Grounding process

the candidate synsets for each term in table 4.4, and then picked the synset which represents the meaning most related to the programming domain. We found suitable synsets for all the terms except for *coding*. The grounding of terms is presented in table 4.5. For each synset we create an RDF resource following the W3C recommendations in this respect[1]. A URI for an RDF resource representing a synset has to adhere to the pattern http://www.w3.org/2006/03/wn/wn20/instances/synset-[Name]-[POS]-[number] where *Name* is the synset name, *POS* is the part of speech, and *number* is the synset number which identifies the meaning of the word. Thus the output of this activity is the stream of strings representing RDF resources presented in table 4.6. In this table resources have been prefixed with the *wn* namespace which refers to http://www.w3.org/2006/03/wn/wn20/.

[1]RDF/OWL Representation of WordNet: http://www.w3.org/TR/wordnet-rdf/

Table 4.5: Grounding of terms to senses in WordNet

Term	**Synset**	**Description**
Java	java#n#3	A simple platform-independent object-oriented programming...
program	program#n#7	(computer science) a sequence of instructions that a computer...
programming	programming#n#2	Creating a sequence of instructions to enable the...
language	programming_language#n#1	(computer science) a language designed for programming...
open-source	open-source#a#1	Of or relating to or being computer software for...

Table 4.6: RDF resources representing WordNet synsets

wn:instances/synset-java-noun-3
wn:instances/synset-program-noun-7
wn:instances/synset-programming-noun-2
wn:instances/synset-programming_language-noun-1
wn:instances/synset-open_source-adverb-1

4.3.2 Activity 2.2. Identification of classes

The Identification of Classes activity aims at obtaining classes in the knowledge base that correspond directly or are related to the semantic entities gathered by the Semantic Grounding activity. Each semantic entity may be connected in the knowledge base to other entities which may represent classes. These connections between entities and classes can span through different entities and therefore we have to be able to browse them so that we can obtain the related classes.

Guidelines. The definition of semantic entities given above allows them to be ontology classes and hence these classes may be collected first. Next step is to follow the relations between semantic entities and classes that are defined in the knowledge base. Specifically, we are interested in relations having as domain a class and as range the semantic entity, or the opposite, relations having as domain the semantic entity and as range a class. Note that the relation may not be direct since they can involve intermediate entities. Ontology engineers must define the relations they want to benefit from to obtain the domain relevant classes. Examples of these relations are equivalence relations such as *owl:sameAs* or *owl:equivalentClass*, relations defining instances of classes *rdf:type*, relations defining subclasses *rdfs:subClassOf*, or ad-hoc relations which are important in the domain of study.

Some of the classes obtained from the semantic resources may be too general to be considered as relevant in the domain. The level of generality of a given class can be measured by calculating the distance from the class to the root class in the knowledge base hierarchy (Wu and Palmer, 1994). Usually this distance is measured as the number of relations or of classes that define a path between the two classes. A short distance is an indicator of generality while a long distance is an indicator of specificity.

Since the Semantic Elicitation process can benefit from more than one knowledge base, **in this activity we can obtain many classes that are equivalent through different knowledge bases**, and thus ontology engineers must decide how to deal with those equivalent classes. Current modeling languages and reasoners allows representing and infering over equivalent classes. Nevertheless, from the user point of view equivalent classes can hamper the understanding and maintainability of the knowledge base. We have identified three possible alternatives as to how to include this equivalent classes into the new ontology. The first one is to keep all the equivalent classes, thus sacrificing the ontology understanding by users. The second alternative is to keep one of the equivalent classes and discarding the others. In this alternative some valuable information can be lost with the discarded classes. The third option is to create a unique class to represent each group of equivalent classes. These unique classes state the equivalence with the existing classes using the relation *owl:equivalentClass*, and they may also aggregate the data properties of the equivalent classes. This last option offers a good solution from the user point of view while preserving the source information so that reasoning capabilities are maintained.

In section 5.2.2 we propose a technique to identify classes, in a linked data set, from semantic entities. The technique searches for all the possible paths connecting, through *sameAs* relations, a semantic entity and a class. The search in the linked data set is done via SPARQL queries. In addition in section 6.3 we have evaluated this technique in the context of an experiment to develop a stock market ontology. In this experiment the semantic entities belong to DBpedia and the classes belong to a linked data set consisting of DBpedia, OpenCyc, and UMBEL ontologies. In addition to represent the classes we decided for the third option discussed above which is to create classes in the ontology and linked them with the existing classes in the linked data set.

Formalization. In listing 4.12 we present the definition of the Identification of Classes activity in DISPEL. We define a processing element type *IdentifyClasses*. The input is defined by the stream of semantic entities produced in the Semantic Grounding. The output is a stream of classes represented by triples of *rdf:Statement* type. The details of how a class is defined using an RDF statement are defined in the specification of the language in which we are creating the ontology. A processing element of type *IdentifyClasses* is created by means of the function *identifyClassesFn*. This function receives as input the knowledge base used for the class identification. The actual process to identify classes is implemented in the program *IdentifyClassesImpl*.

Example. We consider noun synsets as classes to be included in the ontology. However, not all nouns represent classes and thus including them in the ontology is a conceptualization error. This is the case of named entities such as cities, monuments, or people, among other types. For instance, *Paris* the city is an instance and not a class. WordNet distinguishes between classes and instances by classifying nouns in *Types* and *Instances*. Therefore from the RDF resources presented in table 4.6 we only select as classes those noun synsets that are Types. Then we create the RDF triples defining the classes. For each class we also define its labels which correspond to the different synonyms associated to the synset. The output stream is shown in table 4.7. Note that class definition also include the labels for each class which have been extracted from the synonyms associated to each synset.

4.3.3 Activity 2.3. Relation discovery

The final activity proposed in our method is the discovery of relations between classes. The goal of this activity is twofold: 1) to gather relations, in the form of object properties, which relate classes identified in the previous activity, and 2) to produce the final domain ontology. To elicit these relations we propose to rely on the knowledge bases where we can search and extract them. The ontology is then created with the set of

```
package es.upm.oeg{
  //Implementation of a program to identify classes
  use es.upm.oeg.IdentifyClassesImpl;

  //Processing element type to identify classes
  Type IdentifyClasses is
    PE(<Connection:String::"rdf:Resource" entity>
       => <Connection:Statement::"rdf:Statement" classDefinition>);

  //Constructor of the processing element of type IdentifyClasses
  PE(IdentifyClasses) identifyClassesFn(String knowledgeBase){
    //Instance of the program to identify classes
    IdentifyClassesImpl identifyClassesImpl=new IdentifyClassesImpl();

    //Set the knowledge base to be used
    |-knowledgeBase-|=>identifyClassesImpl.inputKnowledgeBase;

    //Connects the stream of semantic entities to the input of the program
    //and the output of the program to the stream of class definitions.
    return(<Connection entity=identifyClassesImpl.inputEntity>
           => <Connection classDefinition=identifyClassesImpl.outputClassDef>);
  }
}
```

Listing 4.12: Definition of the Processing Element Type for the Identification of Classes activity

classes, their definitions and the relations among them. Thus, the input of this activity is the stream of classes and their definitions produced by the Class Identification activity. The output is a list of statements of the form *subject predicate object* representing the Ontology. The relations discovered between two classes can span through some unidentified classes, and thus these new classes may be added to the ontology. Therefore this activity is recursive in nature since newly identified classes arising from the discovered relations can be used in a new search for relations among the new and the existing classes.

Guidelines. General purpose knowledge bases can contain generic relations which may not fit the requirements of domain ontologies. Examples of these relations are *Open-*

Table 4.7: Definition of classes for the resources identified in the semantic grounding

<wn:instances/synset-java-noun-3> <rdf:type> <rdfs:Class>.
<wn:instances/synset-java-noun-3> <rdfs:label> "Java"@en.
<wn:instances/synset-program-noun-7> <rdf:type> <rdfs:Class>.
<wn:instances/synset-program-noun-7> <rdfs:label> "program"@en.
<wn:instances/synset-program-noun-7> <rdfs:label> "programme"@en.
<wn:instances/synset-program-noun-7> <rdfs:label> "computer program"@en.
<wn:instances/synset-programming-noun-2> <rdf:type> <rdfs:Class>.
<wn:instances/synset-programming-noun-2> <rdfs:label> "programming"@en.
<wn:instances/synset-programming-noun-2> <rdfs:label> "programing"@en.
<wn:instances/synset-programming-noun-2> <rdfs:label> "computer prog.."@en.
<wn:instances/synset-programming_language-noun-1> <rdf:type> <rdfs:Class>.
<wn:instances/synset-programming_language-noun-1> <rdfs:label> "prog.."@en.
<wn:instances/synset-programming_language-noun-1> <rdfs:label> "prog.."@en.

Cyc:QouteIsa[1] which is a relaxed version of the relation *rdf:type*, and *dcterms:subject*[2] which is used in DBpedia to relate resources to Wikipedia categories. Hence, **the ontology engineer can define a *whitelist* with the relations they are interested in, or otherwise a *blacklist* for the relations they are not interested in**.

As we mention before we are interested not only in direct relations between the classes, but also in indirect relations spanning other classes. The **classes found in-between of indirect relations are possible related to the domain of study, though this level of relation may be influenced by two factors**. The first factor is the number of unidentified classes making up the relation between the classes for which the relation was searched. A small number of unidentified classes can suggest that there is a strong relation between them and the domain of study, while a large number of these classes may indicate the opposite. The second factor is the specificity level of the unidentified classes. A relation between to classes, which span through general classes such as *thing* or *abstract entity*, can suggests a low relevance of the relation to the domain. The specificity level can be measure in an ontological hierarchy by the class distance to the root concept. A short distance to the root indicates a low

[1]http://sw.opencyc.org/concept/Mx4rBVVEokNxEdaAAACgydogAg
[2]http://purl.org/dc/terms/subject

level of specificity while a long distances suggest a high level of specificity. Thus ontology engineers must define the guidelines to accept the relations and new classes discovered in terms of the number of classes which make up the relation and the specificity of these classes.

In addition **ontology engineers have to take into account the computational cost of searching and retrieving the relations for a given pair of classes from the knowledge base**. In large knowledge bases this search can be very expensive, in terms of processing time and machine resources, since usually it implies to traverse the whole knowledge base. Although the search performance depends, in a large extent, of the knowledge base implementation and of the machine capabilities over which it is deployed, some strategies can be designed to limit the scope of the search and help the knowledge base server in this task. These strategies are framed in what is known as query, knowledge base, and server tunning, and they varies widely between the different query languages, knowledge base implementations and operating systems, and hence we suggest ontology engineer to follow the specific tunning guidelines of each of the aforementioned components.

Finally **the decisions taken in the previous activity regarding how to deal with equivalent classes influences how to represent the relations in the final ontology**. If the option was to keep all the equivalent classes or just one of them, then the relations discovered during this activity have to be defined for the corresponding classes. If the option was to create local classes linked to the existing equivalent classes then the relations discovered have to be defined between the local classes.

In section 5.2.3 we present a technique to identify relations among classes in a linked data set. This technique is based on issuing SPARQL queries to traverse all the possible paths connecting to classes in the linked data set. In practice the length of the maximum path to traverse has to be defined. We have evaluated this technique in section 6.3 within the framework of a process to develop an ontology in the sotck market domain. In this experiment the linked data set consisted of DBpedia, OpenCyc and UMBEL ontologies. The identified relations were created in the ontology as relations between the ontology classes.

Formalization. In listing 4.13 we present the definition in DISPEL of the processing element type for the Relation Discovery activity. The input is the stream of classes and their data properties, which are represented by the *rdf:Statement* structural type.

The processing element type defining this process is *DiscoverRelations*. The output is the ontology itself represented by a stream of *rdf:Statement* variables. Valid statements depends on the ontology language and they include the definition of a class, an object property, a data property, and the association of a property to another property or to a class. In this last case when an object property links to classes a relation is established. The constructor of a processing element of type *DiscoverRelations* is the function *discoverRelationsFn*. This function receives as input data the knowledge base used for discovering the relations. The actual implementation of the process is the program *DiscoverRelationsImpl*.

```
package es.upm.oeg{
  //Implementation of a program to discover relations
  es.upm.oeg.DiscoverRelationsImpl;

  //Processing element type for the process to discover relations
  Type DiscoverRelations is
  PE (<Connection:Statement::"rdf:Statement" classDefinition>
      => <Connection:Statement::"rdf:Statement" ontologicalStatement>);

  //Constructor of a processing element of type DiscoverRelations
  PE(DiscoverRelations) discoverRelationsFn(String knowlegeBase){
    //Instance of the program to discover relations
        DiscoverRelationsImpl discoverRelationsImpl= new DiscoverRelationsImpl();

        //Set the knowledge base used for discovering relations
        |-knowlegeBase-| => discoverRelationsImpl.inputKnowledgeBase;

        //Connect the stream of class definitions to the input stream of the program
        //and the output stream of the program to the stream of rdf statements
        return (<Connection classDefinition = discoverRelationsImpl.inputClassDef>
             => <Connection ontologicalStatement=discoverRelationsImpl.ouputRDFStatements);
  }
}
```

Listing 4.13: Definition of the Processing Element Type for the Relation Discovery activity

Example. For the discovery of relations we use the WordNet hyponymy hierarchy. We translate hyponymy relations between noun synsets, representing types, as *subClassOf* ontological relations. First we carry out a pairwise search for paths in the hyponomy hierarchy linking the synsets corresponding to the classes found in the previous activity (see table 4.7). Results are shown in table 4.8. For each pair of synsets we present the path found in the hyponym hierarchy. Note that whenever the path is established through a common superclass in the hierarchy we show the corresponding synset in bold. For instance *program* and *programming_language* share *communication#n#2* as common super class. This means that both synsets are types of communications. Paths including a common superclass have to be interpreted in a different way. That is previous to the common super class the relation between synsets is *Is A* while after it the relation between synset is the inverse of *Is A* (e.g., *Type Of*). All the relations found in Wordnet, except for the relations between *Java* and *programming language*, include a common superclass.

In total we found 16 new classes setting a path between the previously identified classes. Therefore we include them in our final ontology. In addition we create the *subClassOf* relation between the classes according to the hyponomy relations found for each pair of classes. For instance the relation between *java* and *object-oriented_programming* is formalized as follows: <wn:synset-java-noun-3> <rdfs:subClassOf> <wn:synset-object-oriented\programming\language-noun-1>. Thus the output stream of this activity is the ontology itself representing by the triples defining the classes we have identified and the relations between those classes. In figure 4.6 we depict the obtained ontology.

4.3.4 Choreography of the semantic elicitation process

After defining the activities which have to be carried out during the Semantic Elicitation process we can present how these activities are orchestrated in an ordered sequence to produce the domain ontology. In listing 4.14 we describe the function *elicitSemantics* which is in charge of creating the workflow (see figure 4.7). This function creates a processing element of type *SemanticElicitation* which represent the process itself. This process input is a stream of contextualized terms that is connected to the internal connection *inputTerms*. This stream of terms is then fed into the input of *groundProcInst* process. This grounding process carries out the Semantic Grounding activity and obtain a list of semantic entities. Next the stream of semantic entities is sent to the input of the *getClassesProcInst* process which implements the *Identification of Classes* activity. The output of this process is a stream of statements describing the ontology classes and their definitions. These statements are the input of the *getRelationsProcInst* process

Table 4.8: Relation discovery between classes. Synsets in bold indicates common superclass.

Class java and Class program
Java#n#3 object-oriented_programming_language#n#1 programming_language#n#1 artificial_language#n#1 language#n#1 **communication#n#2** written_communication#n#1 writing#n#4 coding_system#n#1 code#n#3 software#n#1 program#n#7
Class Java and Class programming
Java#n#3 object-oriented_programming_language#n#1 programming_language#n#1 artificial_language#n#1 language#n#1 communication#n#2 **abstraction#n#6** psychological_feature#n#1 event#n#1 act#n#2 activity#n#1 creation#n#1 creating_by_mental_acts#n#1 programming#n#2
Class java and Class programming language
Java#n#3 object-oriented_programming_language#n#1 programming_language#n#1
Class program and Class programming
program#n#7 software#n#1 code#n#3 coding_system#n#1 writing#n#4 written_communication#n#1 communication#n#2 **abstraction#n#6** psychological_feature#n#1 event#n#1 act#n#2 activity#n#1 creation#n#1 creating_by_mental_acts#n#1 programming#n#2
Class program and Class programming_language
program#n#7 software#n#1 code#n#3 coding_system#n#1 writing#n#4 written_communication#n#1 **communication#n#2** language#n#1 artificial_language#n#1 programming_language#n#1
Class programming and Class programming_language
programming#n#2 creating_by_mental_acts#n#1 creation#n#1 activity#n#1 act#n#2 event#n#1 psychological_feature#n#1 **abstraction#n#6** communication#n#2 language#n#1 artificial_language#n#1 programming_language#n#1

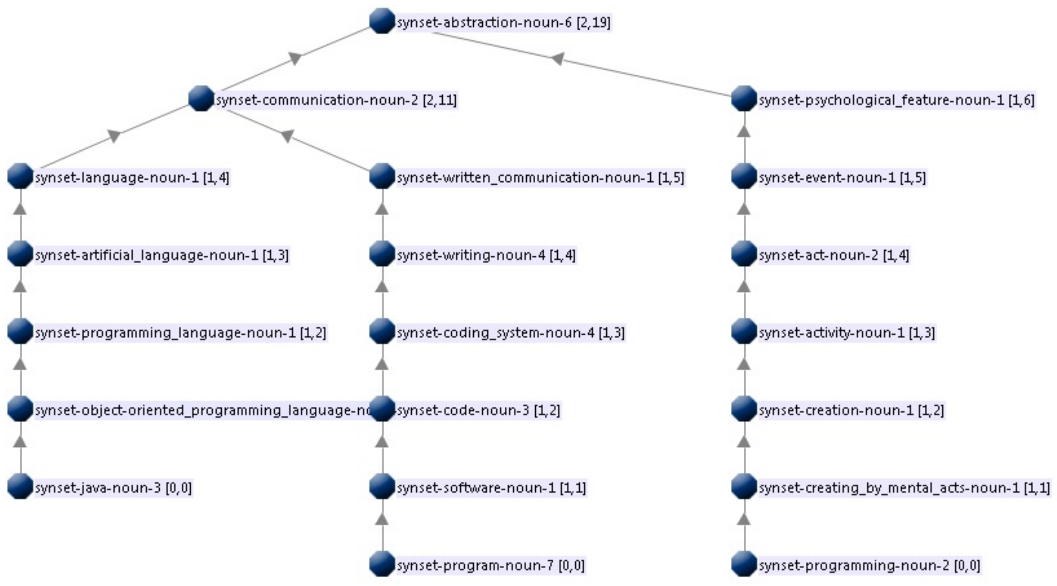

synset-abstraction-noun-6 [2,19]
synset-communication-noun-2 [2,11]
synset-psychological_feature-noun-1 [1,6]
synset-language-noun-1 [1,4]
synset-written_communication-noun-1 [1,5]
synset-event-noun-1 [1,5]
synset-artificial_language-noun-1 [1,3]
synset-writing-noun-4 [1,4]
synset-act-noun-2 [1,4]
synset-programming_language-noun-1 [1,2]
synset-coding_system-noun-4 [1,3]
synset-activity-noun-1 [1,3]
synset-object-oriented_programming_language-no
synset-code-noun-3 [1,2]
synset-creation-noun-1 [1,2]
synset-java-noun-3 [0,0]
synset-software-noun-1 [1,1]
synset-creating_by_mental_acts-noun-1 [1,1]
synset-program-noun-7 [0,0]
synset-programming-noun-2 [0,0]

Figure 4.6: Programming ontology obtained from the classification system and from the reuse of WordNet conceptualizations.

which carries out the Relation Discover activity. This process output is the stream of statements defining the ontology which are delivered through the output of the Semantic Elicitation process.

4.4 Choreography of the method for developing ontologies

In summary the method proposed in this thesis consists of two main processes: the Terminology Extraction process presented in section 4.2, and the Semantic Elicitation process described in section 4.3. Both processes have been described and divided in activities and tasks, and formalized in DISPEL.

The workflow definition (see listing 4.15 and figure 4.8) starts by creating the processing element of the types defined for the *Terminology Extraction* and *Semantic Elicitation* activities. From these types two process are instantiated: *extractTerminologyProcInst* and *elicitSemanticsProcInst*. In addition a *results* process is created which is in charge of displaying the generated ontology. Next, the processes are orchestrated. The stream of terms produced by the *extractTerminologyProcInst* process is connected to the input of the *elicitSemanticsProcInst* process. The *elicitSemanticsProcInst* process produces

```
package es.upm.oeg{
  PE(SemanticElicitation) elicitSemantics(String knowledgeBase)  {
  //Input stream of the Semantic Elicitation process
  Connection inputTerms;

  //Create the processing elements types of each activity
  PE(Ground) Grounding = new groundFn(knowledgeBase);
  PE(IdentifyClasses) ClassIdentification = new identifyClassesFn(knowledgeBase);
  PE(DiscoverRelations) RelationDiscovery = new discoverRelationsFn(knowledgeBase);

  //create instances of the processing elements
  Grounding groundProcInst = new Grounding;
  ClassIdentification getClassesProcInst = new ClassIdentification;
  RelationDiscovery getRelationsProcInst = new RelationDiscovery;

  //connect workflow
  inputTerms => groundProcInst.contextualizedTerm;
  ground.entity => getClassesProcInst.entity;
  getClassesProcInst.classDefinition=>getRelationsProcInst.classDefinition;

  return PE(<connnection contextualizedTerm = inputTerms>
          => <Connection ontologicalStatement = getRelations.ontologicalStatement>); }}
```

Listing 4.14: Workflow for the Semantic Elicitation process

```
package es.upm.oeg{
  //Create the processing element of the types defined for each activity
  PE<ExtractTerminology> TerminologyExtraction = new extractTerminologyFn(UGCS,lexicalResource);
  PE<ElicitSemantics> SemanticElicitation = new elicitSemanticsFn(knowledgeBase);

  //Create instances of the processing elements
  TerminologyExtraction extractTerminologyProcInst = new TerminologyExtraction;
  SemanticElicitation elicitSemanticsProcInst = new SemanticElicitation;
  Results results = new Results;

  //connect workflow
  extractTerminologyProcInst.contextualizedTerms => elicitSemanticsProcInst.contextualizedTerms;
  elicitSemanticsProcInst.ontologicalStatement =>  results.input; }
```

Listing 4.15: Workflow defining the method for obtaining ontologies from user-generated classification systems

the stream of statements defining the ontology. These statements are connected to input of the *results* process.

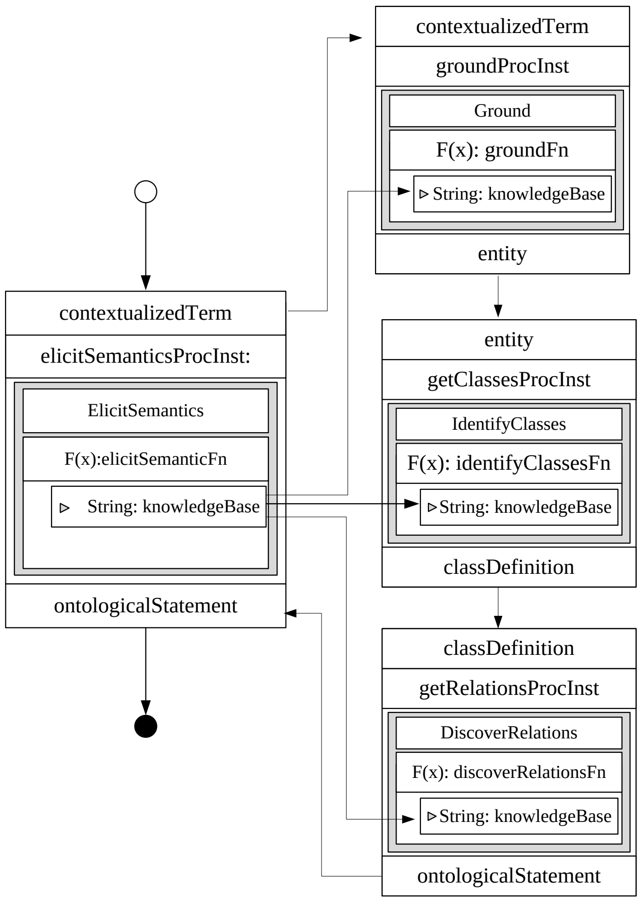


Figure 4.7: Graphical representation of the semantic elicitation process

4.5 Conclusions

In this chapter we have presented our method for developing ontologies from user-generated classification systems. This method was inspired by the activities proposed in well known ontology development methodologies such as Methontology (Fernández-Lopez et al., 1997) and the first scenario of the NeOn methodology (Suárez-Figueroa et al., 2012). For instance the first activity in Methontology is to create a term glossary. We carry out this task in the Terminology Extraction process where we obtain a list of relevant domain terms from the user-generated classification system. Next this

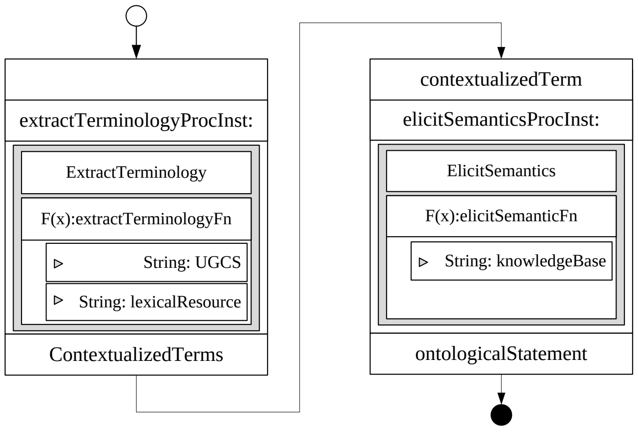


Figure 4.8: Graphical representation of the method for developint ontologies from folksonomies

methodology proposes to create a taxonomy of concepts and then to identify binary relations between those concepts. We carry out these two tasks in the Semantic Elicitation process where we first identify concepts (*i.e.*, classes) from the list of domain terms and then search for relations between those classes in a linked data set. The main difference is that in our method we do not rely on humans to create the ontology but in the knowledge already present in the user-generated classification system and in the ontologies that we are reusing in the process.

Methodologies have evolved from those where experts create the ontologies from scratch (*e.g.*, Methontology (Fernández-Lopez et al., 1997)) to more sophisticated methodologies were it is possible to reuse knowledge from ontological and non-ontological resources (*e.g.*, NeOn methodology (Suárez-Figueroa et al., 2012)). Non-ontological resources are information sources from which it is possible to elicit knowledge, though this knowledge is not yet formalized in an ontology. Thus user-generated classification systems fall into the category of non-ontological resources and they have been recognized as such. In (Villazón-Terrazas, 2012) authors propose patterns and techniques to reuse and reengineering different types of non-ontological resources such as taxonomies, and thesauri. However they do not propose an approach to develop ontologies from user-generated classification systems. In addition from the review of the state of the art we can state that none of the surveyed research works propose an integrated method to generate a domain ontology.

We have formalized our method using DISPEL. This formal language allows describing how the processes are organized and how data are transformed through them. In addition DISPEL allows defining the semantics of the data by means of the association of the data streams with ontological concepts. These semantics descriptions of the data improve the understanding of the workflow by humans and machines.

In chapter 5 we present some techniques to support the main processes and activities proposed in our method. In addition in chapter 6 we present an experiment where we applied the method and the techniques to develop an ontology in the market-value domain from Delicious data and reusing linked data sets.

CHAPTER 5

TECHNIQUES SUPPORTING THE METHOD

In this chapter we describe a set of techniques that support the implementation of our method for developing ontologies from user-generated classification systems. In this context a technique is understood as a specific procedure which can be represented by a sequence of instructions arranged in an algorithm (see section 4.1.1). We propose to use existing techniques tailored to the needs of each activity. That is, we adapt techniques that may have been initially created to target a different problem so that they can fulfill the requirements of the activities and tasks defined in the method. Thus our contributions are to adapt these techniques to new uses which are distinct from the ones on which they are applied traditionally, and to arrange them in an organized sequence so that they contribute to the final goal of developing an ontology from user-generated classification systems.

In figure 5.1 we show the techniques that we have identified to carry out the processes of the method for building ontologies. In the Extract Terminology process we define techniques for: i) the Normalization task of the Preprocessing activity, and ii) the Term Selection activity. For the **Normalization** task we use an **approximate matching** (Zobel and Dart, 1995) technique (see section 5.1.1). The most common application of approximate matchers are in spell checking where words are compared to a list of correct words. If a word does not appear in this list then the most similar words are retrieved as suggestions. In our case we have a list of normalized words which is a large dictionary created from Wikipedia article titles and redirection page titles. Then we apply the approximate matcher to obtain candidate normalizations which are ranked

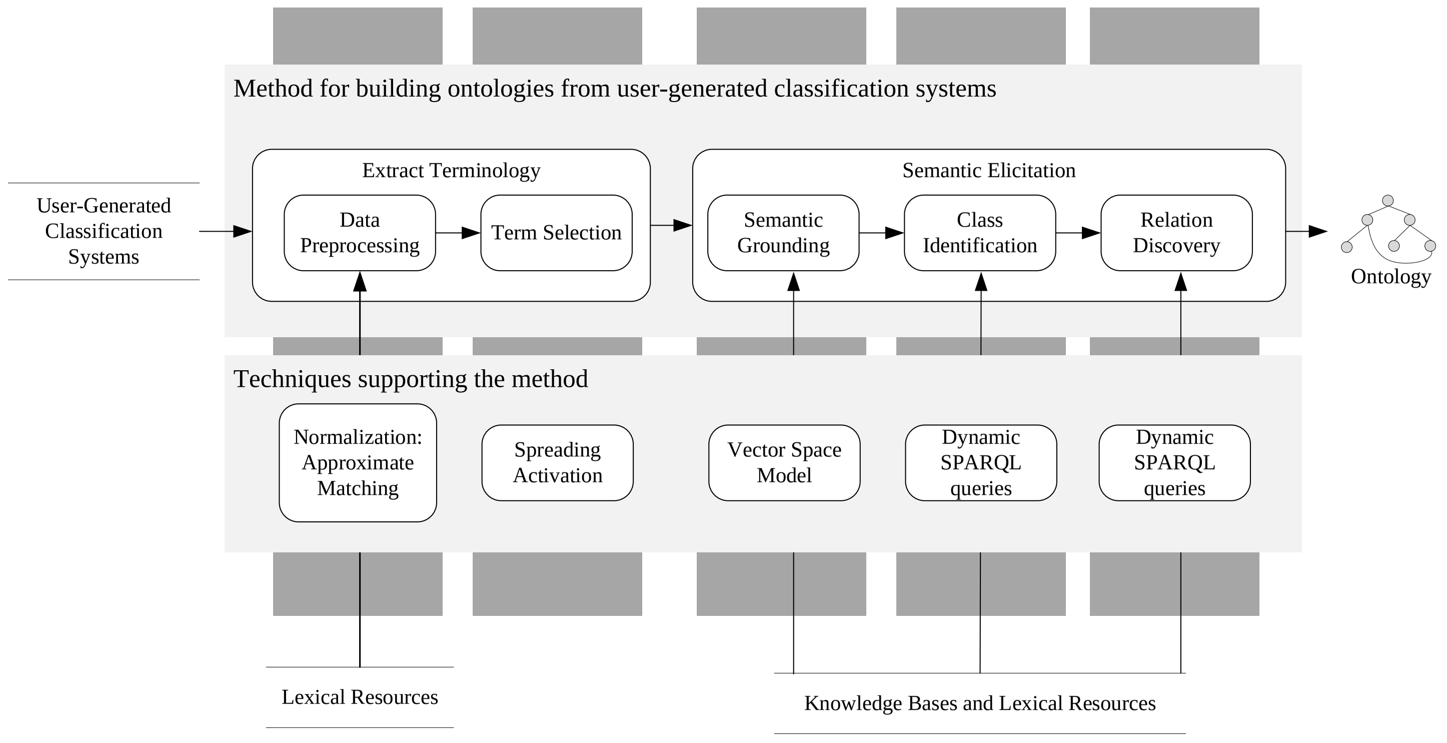


Figure 5.1: Techniques supporting the method for building ontologies.

according to string similarity and popularity. We measure popularity as the number of times that a word has been used in Wikipedia.

For the **Term Selection** activity (see section 5.1.2) we use **spreading activation** (Crestani, 1997) to collect relevant domain terms. Spreading activation is used to search networks of any type and it has been used in information retrieval processes (Crestani, 1997) where the network consists of nodes which represent the documents and terms in those documents. In our case the network is a graph representing a part of the user-generated classification systems. Then we traverse this graph, starting from a set of predefined nodes, using spreading activation so that we can collect the category names of the activated nodes as domain relevant terms. We have defined an activation function, which is applied on each visited node, based on the number of shared annotations between the previously visited node and the current node.

In the Semantic Elicitation process we define techniques for all the activities. For the **Semantic Grounding** activity we use the **vector space model** (Salton and Mcgill, 1986) to select the concept which represents the meaning of a category name (see section 5.2.1). The Vector space model has been used traditionally in information retrieval processes (Baeza-Yates and Ribeiro-Neto, 2011); it allows representing documents as vectors of terms which then can be compared with a query vector to select the most similar document vectors. In the semantic grounding we consider the category name and its context as a query and the candidate semantic entities as documents, from which we have to select the one that better represents the category name meaning. Thus we create a query vector with the category name and its context and look for the most similar sense in a candidate set of senses. The candidate set of senses is created from DBpedia disambiguation resources where the possible concepts representing the meanings of words are specified.

For the **Class Identification** and **Relation Discovery** activities (see sections 5.2.2 and 5.2.3) we use **dynamic SPARQL queries** (Prud'hommeaux and Seaborne, 2008). Dynamic generation of SPARQL queries has been used in (Heim et al., 2010), in the context of reusing knowledge from ontologies and linked data, to find relations between two existing resources. In the Class Identification activity we use this technique to traverse all the possible paths, in the RDF graph defined by DBpedia, OpenCyc and UMBEL ontologies, connecting a given resource, via *sameAs* links, to a class. In the

Relation Discovery activity we use this technique to search for relations between classes in the same RDF graph.

Along with the descriptions of each of these techniques we present the algorithms in pseudocode. Please note that the procedures and functions of each algorithm are presented in order of dependency. Non-dependent procedures and functions appear on top of the algorithm description while dependent ones are at the bottom. When describing each algorithm we make clear the main procedure from which the reading of code can be started.

5.1 Terminology extraction

The extraction task in the preprocessing activity depends on the data source and on how the data are exposed by the owner, and hence is not feasible to propose a unique technique to carry out the extraction. Nevertheless, many web-based systems have adopted RESTful web services (Richardson and Ruby, 2007) to allow querying and updating their data. Thus, regardless of how the classification instances were extracted, they have to be stored in an initial stage area (*e.g.*, database tables, XML documents or flat files) from which the next tasks can query the data.

Continuing with the preprocessing activity, for the normalization task we propose to use a technique (see section 5.1.1) to search over a dictionary using n-grams and string similarity measures so that category names can be compared and normalized to the dictionary entries. Note that we are not providing a technique for the transformation task in the data preprocessing activity since the transformation details depend on the storage system selected for the data. For instance for relational databases the transformation might be defined in terms of SQL queries (Chamberlin and Boyce, 1974) and of sentences in the data definition language, while for XML documents the transformation might involve the use of XSL Transformations (Clark, 1999). Nevertheless the target schema to which data are transformed depends entirely on the information needs of the term selection activity. Given that we are also proposing a technique for the term selection activity (see section 5.1.2) we include a description of the input data schema, which is the output of the transformation task, required to carry out the activity. The technique used for the term selection activity is based on a spreading activation strategy over a graph created from the transformation of the classifications instances.

5.1.1 Approximate matching for normalization

We propose to use a dictionary in the form of a list of words so that category names can be compared with entries in the dictionary. Category names found in the dictionary using exact match string comparison, are considered as normalized words. Category names not found in the dictionary are subjected to an approximate matching process with the dictionary entries. An approximate string matching process takes as input a query string and produces as output approximate matches of entries in a lexicon or dictionary (Zobel and Dart, 1995). A survey of approximate string matching techniques is presented in Navarro (2001).

To solve this problem we follow the architecture proposed in Zobel and Dart (1995) which consists of a *coarse search* of candidate entries in the lexicon, and of a *fine search* to narrow the selection of the right candidates. We have a selected a pragmatic approach (White, 2005) to address this problem, which is able to deal with large lexicons since it uses the vector space model (Salton and Mcgill, 1986) for retrieving the candidate matching entries.

The approximate matching process proposed in White (2005) is based on the comparison of the n-grams of a category name and of the dictionary entries. An n-gram is a contiguous sequence of n items taken from a unit of text. Thus an item can take the form of a character or a word. Thus, for grams of size 1 (1-grams) we have one item, for grams of size 2 (2-grams) we have two items, and so forth. For instance, table 5.1 presents n-grams extracted from the category name *SemanticWeb* with characters as items and n values ranging from 1 to 4.

We carry out a coarse search, based on n-grams of the dictionary entries, to obtain an initial set of candidates (Zobel and Dart, 1995). That is, the n-grams of dictionary entries are compared to the n-grams of a category name so that we can obtain those entries which are more similar to the category name. To implement an efficient search we use the vector space model.

Each entry in the dictionary is represented as a vector in $\Re^k$ where k is the cardinality of the set created from the union of all n-grams appearing in the entries. Each position in the vector corresponds to a given n-gram and its value can be defined according to any weighting scheme including term frequency (TF) or term frequency-inverse document frequency (TF-IDF), though in this case terms are n-grams. Once we have created all

Table 5.1: N-grams for the SemanticWeb category name

1-gram	2-gram	3-grams	4-grams
S	Se	Sem	Sema
e	em	eman	eman
m	ma	man	mant
a	an	ant	anti
n	nt	nti	ntic
t	ti	tic	ticW
i	ic	icW	icWe
c	cW	cWe	cWeb
W	We	Web	
e	eb		
b			

the vectors for the entries we can use the vector space for approximate matching as follows. For a category name its n-grams are extracted, and a vector in the same space ($\Re^k$) is created. Each position of this vector is assigned a weight of 1 for those n-grams appearing in the category name and of 0 otherwise. Next, we compare the category name vector with each entry vector by calculating the cosine of the angle formed by both vectors. The greater the cosine of the angle, the higher the similarity between the vectors. Thus we retrieve the set of most similar entries to the category name.

Finally, during the fine search we compare, using a string similarity measure, the category name with each of the entries so that we can obtain the most similar one. Note that string similarity measures can rely on edit distances such as Levenshtein's or on token based distances such as Jaccard similarity[1]. Besides the string similarity we also take into account the popularity of the entries so that most popular entries have more opportunities to be selected. The popularity of a category name is measured as the number of times that the corresponding term appears in the dictionary definitions.

Example. In the coarse search we retrieve from our dictionary the following entries for the *SemanticWeb* category name: *Semantic_ Web, Semantic_net, Semantic, Semantic_ MW, and Semantics*. Then we apply Levenshtein's distance between the cat-

[1]For a review of this kind of measures the reader is referred to Cohen et al. (2003)

egory name and each one of the entries. Levenshtein's distance measures the minimum number of characters that have to be inserted, deleted or replaced to convert one string in another. In our case the smaller distance is 1 and corresponds to the *Semantic_ Web* entry which is selected as the normalized version of the category name. Please note that we have postponed the exemplification of the vector space model since later, in section 5.2.1, we show an example of how it works.

5.1.1.1 Algorithm

In algorithm 1 we present the pseudocode for the normalization technique. This algorithm presents a procedure *createVectorSpace* which creates the vector space representation for the dictionary entries according to their n-grams. The normalization is carried out by the *normalize* function, which receives as input the string to normalize and returns its normalized version according to the most similar dictionary entry. The approximate matching is implemented through two functions *coarseSearch* and *fine-Search*. The former searches in the vector space for candidates similar to the input string, while the latter selects the most similar of the candidates according to string similarity and popularity.

Algorithm 1 Normalizing by approximate matching to dictionary entries

```
                                                  ▷ Initialize global variables.
 1: $allNGrams \leftarrow getAllNGrams(dict)$            ▷ Array of all dictionary n-grams.
 2: $vectorSpaceMatrix \leftarrow Matrix[length(dict)][length(allNgrams)]$
                   ▷ Create the matrix for the vector space (dic. entries X ngrams)
 3: procedure CREATEVECTORSPACE($dict$)
 4:     $i \leftarrow 0$
 5:     for all $entry \in dict$ do
 6:         $j \leftarrow 0$
 7:         for all $gram \in allNGrams$ do
 8:             $VectorSpaceMatrix[i][j] \leftarrow calcTFIDF(dictionary, entry, gram)$
 9:             $j \leftarrow j + 1$
10:         end for
11:         $i \leftarrow i + 1$
```

```
        end for
end procedure
                                        ▷ Create the vector representing the string to normalize
function createQueryVector(strToNormalize)
    $queryVector \leftarrow Array[length(allNgrams)]$
    $ngrams[] \leftarrow getNgrams(strToNormalize)$
    $i \leftarrow 0$
    for all $gram \in allNGrams$ do
        if $gram \in ngrams$ then
            $queryVector[i] = 1$
        else
            $queryVector[i] = 0$
        end if
        $i \leftarrow i + 1$
    end for
    return $queryVector$
end function
                              ▷ Select from the vector space the most similar entry to the string
function coarseSearch(strToNormalize)
    $queryVector \leftarrow getQueryVector(strToNormalize)$
    $candidates \leftarrow Array[length(dict)]$
    $i \leftarrow 0$
    for all $entry \in dict$ do
        $entryVector \leftarrow VectorSpaceMatrix[i]$
        $simValue \leftarrow calcCosine(queryVector, entryVector)$
        $candidates[i] \leftarrow entry$
        $setSimValue(candidates[i], simValue)$
        $i \leftarrow i + 1$
    end for
    $sort(candidates, \text{“desc”})$      ▷ sort candidates by simValue in descending order
    return $topN(candidates)$                   ▷ Return the most similar candidates
end function
                              ▷ Select the most similar candidate entry using string similarity
```

```
function fineSearch(queryString,candidates)
    $i \leftarrow 0$
    while $i <= length(getRows(candidates))$ do
        $strSimilarity \leftarrow calcStringSimilarity(queryString, candidates[i][0])$
        $popularity \leftarrow calcPopularity(candidates[i][0], dict)$
        $setStrSimilarity(candidates[i], strSimilarity)$
        $setPopularity(candidates[i], popularity)$
        $i \leftarrow i + 1$
    end while
    $sort(candidates, \text{“desc”})$        ▷ sort candidates by strSimilarity and popularity
    return $firstElement(candidates)$        ▷ Return the most similar candidate
end function
                                             ▷ Main function: normalize strToNormalize
Require: $createVectorSpace(\text{dict})$
function normalize(strToNormalize)
    if $strToNormalize \in dict$ then        ▷ if the string is a dictionary entry
        return $strToNormalize$
    else        ▷ otherwise use approximate matching to find a dictionary entry
        $candidates \leftarrow coarseSearch(strToNormalize)$        ▷ Search using n-grams
        $candidate \leftarrow fineSearch(strToNormalize, candidates)$        ▷ Search str. sim.
        return $candidate[0]$        ▷ return the most similar entry as normalization
    end if
end function
```

5.1.2 Spreading activation for term selection

The technique used for term selection relies on a graph representation of a part of the classification system information. Classification systems can be represented as a stream of classification instances (see listing 4.1), or alternatively as a tripartite hyper-graph[1] $G = (V, E)$ where $V = U \cup CN \cup R$, $E = \{(u, cn, r) | (u, cn, r) \in CI\}$, and U, CN and, R are finite sets whose members are users, category names and resources respectively. Recall that CI was defined previously in chapter 4 as a subset of $U \times CN \times R$. This

[1]A graph is tripartite when the nodes can be divided in three disjoint sets and nodes in the same set are not adjacent.

graph representation was first introduce by Mika (2007) for folksonomies and he showed how this triparte graph can be reduced to two and one mode graphs.

We are particularly interested in the one mode 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 category name assigned to both resources regardless of the user. Formally, $E' = \{(r_i, r_j) | \exists((u, cn_m, r_i) \in CI \wedge (u, cn_n, r_j) \in CI \wedge cn_m = cn_n)\}$[1]. Vertices of the graph have as attributes the list of category names under which they have been classified along with the classification frequency, that is the number of times that users have assigned the category name to the resource. We have chosen this graph because previous studies on folksonomies such as Golder and Huberman (2006) showed that tags tend to converge around resources.

Therefore a prerequisite to apply the technique described in this section is that the normalized classification instances are converted, in the transformation tasks (see section 4.2.1.3), into tuples which define the edges in the graph G' and the attributes of each one of the edge vertices. Please note that for the attributes we require the normalized versions of category names as opposed to their original versions since they do not provide any added value to the rest of the process. Thus classification instances have to be converted into tuples of the form $< r_i, r_j, [< cn_i, freq_i >], [< cn_j, freq_j >] >$ where r_i and r_j are adjacent vertices in G' and the lists of tuples represent the attributes (normalized category name and frequency) of each vertex.

To collect the terms we use spreading activation (Crestani, 1997), a technique that allows taking decisions over each vertex visited in the graph so that we can gather the category names associated with each vertex which are relevant to the domain. Spreading activation is a graph search method initiated by a set of seed nodes[2] weighted with an activation value. Each seed activation value spreads through the linked 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. To traverse the graph we use a breadth-first search (BFS) strategy.

Seeds are the starting point of the spreading activation, and in our case they play an important role, that of being a source of domain relevant category names. Thus we

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

[2]Note that vertices and nodes are synonyms in this context.

require that domain experts provide as input a set of resources $S \subset R$ considered highly pertinent to the domain. We can compare resources connected in G' with a seed in terms of shared category names; category names of highly similar resources are added to the domain term list. Similarity between resources is calculated by the activation function. Next we follow the edges of the visited nodes and compare the target nodes, again using the activation function, with the source nodes. The process continues until the activation value is under a threshold.

The activation value for a node r_j (see equation 5.1) is calculated by estimating the rate of shared category names with r_i, being r_i the previous visited node from which we reached r_j.

$$a'(r_j) = \frac{|\{cn \in CN|(u, r_j, cn) \in CI\} \cap \{cn \in CN|(u, r_i, cn) \in CI\}|}{|\{cn \in CN|(u, r_i, cn) \in CI\}|} \tag{5.1}$$

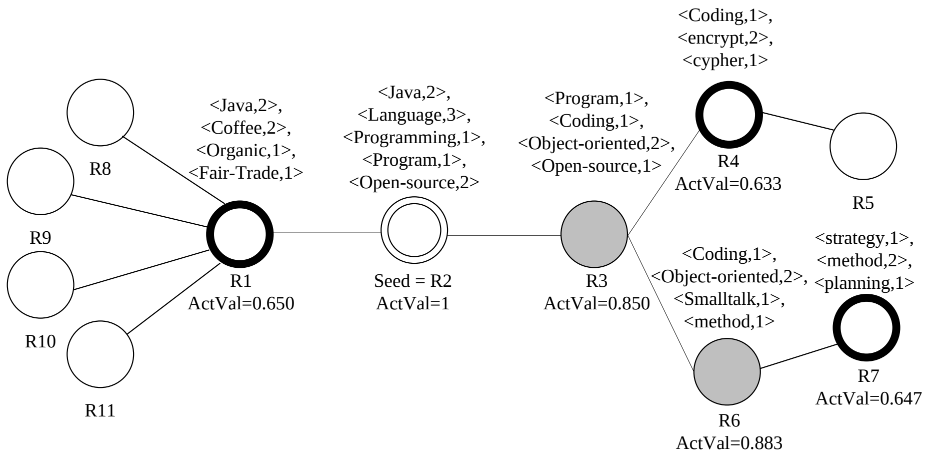
The activation function also depends on the activation value of the previously activated node r_i. Thus, $a(r_j) = a'(r_j) + a(r_i) * \lambda$ where $0 \leq \lambda \leq 1$ is a real number representing a decay factor. If $a(r_j)$ is greater than a threshold h, then it is marked as activated and the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

Once all the seeds are processed we calculate weights for the category names of those activated nodes. We multiply the frequency of each category name by the node activation value (see equation 5.2). Then we gather all distinct category names used to classify the activated nodes and added up their weights. Finally the list of category names is sorted in descending order according to the aggregated weight. As output of this task we created a set of terms from the list of category names representing a valid domain terminology.

$$w(r_j, cn_k) = |\{(u, r_j, cn_k)|(u, r_j, cn_k) \in CI\}| \times a(r_j) \tag{5.2}$$

Example. To exemplify this technique we use the graph presented in figure 5.2 which is an extended version of the user-classification system described in table 4.1. In this graph nodes are resources. Resource names and activation values are placed below the node, while category names and classification frequency are above.

Let us suppose that a domain expert defined R2 as a prominent resource in the domain so that we can use it as a seed. In addition we define a decay factor λ of 0.45, and a threshold h of 0.7. First we visit R2 adjacent nodes R1 and R3 and calculate the activation value. R1 shares one category name with R2, and thus the activation value is $a(R1) = 1/5{+}1{*}0.45 = 0.650$. Given that $a(R1) < h$ we stop the search in this direction. On the other hand, R3 shares two category names with R2 and hence the activation



<Coding,1>, <encrypt,2>, <cypher,1>
<Java,2>, <Coffee,2>, <Organic,1>, <Fair-Trade,1>
<Java,2>, <Language,3>, <Programming,1>, <Program,1>, <Open-source,2>
<Program,1>, <Coding,1>, <Object-oriented,2>, <Open-source,1>
R4
ActVal=0.633
R5
R8
R9
R1
ActVal=0.650
Seed = R2
ActVal=1
R3
ActVal=0.850
<Coding,1>, <Object-oriented,2>, <Smalltalk,1>, <method,1>
<strategy,1>, <method,2>, <planning,1>
R10
R11
R7
ActVal=0.647
R6
ActVal=0.883

Figure 5.2: Spreading activation over a graph of resources using R2 as a seed, a decay factor of 0.45 and threshold of 0.7. Nodes with border in bold are those where the spreading stop. Activated nodes are filled with gray.

value is $a(R2) = 2/5 + 1.0 * 0.45 = 0.850$. Since $a(R3) \geq h$ we activate the node and collect its associated category names. Then we visit R3 adjacent nodes R4 and R6 and calculate their corresponding activation values: $a(R3) = 1/4 + 0.850 * 0.45 = 0.633$ and $a(R6) = 2/4 + 0.850 * 0.45 = 0.833$. We stop the search in R3 direction ($a(R3) < h$) while we activate R6 ($a(R6) \geq h$) and collect its category names. Next we visit R6 adjacent node R7 and calculate its activation value $a(R7) = 1/4 + 0.833 * 0.45 = 0.647$ where we finalize the spreading activation since $a(R7) < h$ and it is the last node to visit.

After finalizing the spreading activation we calculate the weights of each of the category names belonging to the activated nodes R2, R3 and R6 (see table 5.2). Next we group the category names regardless the resource, added up their weights, and sort them in descending order according to the weights. The final list of categories is presented in table 5.3. Hence, if we define a lower limit of 1.0 we discard *smalltalk* and *method*, and keep the rest of category names as the output list of terms.

5.1.2.1 Algorithm

Algorithm 2 consists of a main function *collectTerms* which uses breath first search *BFS* and spreading activation over the graph. BFS is in charge of traversing the graph G' starting from an user-defined seed. Recall that G' is a graph whose vertices are resources, and there is an edge between two resources if they share at least a category name. Vertices have as attributes the category names under which they have been classified.

Table 5.2: Weights per category

Res.	Category Name	Weight
R2	Java	2.000
	Language	3.000
	Programming	1.000
	Program	1.000
	Open-Source	2.000
R3	Program	0.850
	Coding	0.850
	Object-Oriented	1.700
	Open-Source	0.850
R6	Coding	0.883
	Object-Oriented	1.765
	SmallTalk	0.883
	method	0.883

Table 5.3: Aggregated weights per category

Category Name	Sum(Weight)
Object-Oriented	3.465
Language	3.000
Open-Source	2.850
Java	2.000
Program	1.850
Coding	1.733
Programming	1.000
SmallTalk	0.883
method	0.883

For each visited vertex the activation value is calculated using the *activationFunction*. We collect category names as relevant terms for those vertices with an activation value over a predefined threshold. The activation value depends on the shared categories between the vertices, and on the activation value of the source vertex from which we reached the current vertex.

Algorithm 2 Collecting domain relevant terms with spreading activation

```
          ▷ Breadth first search over the G' graph using s as a seed to drive the search
 1: function BFS($G'$, $s$)
 2:     $actVertices \leftarrow Array[]$                          ▷ Array of activated vertices
 3:     $q \leftarrow Queue()$
 4:     $enqueue(q, s)$
 5:     $setState(s, \text{“visited”})$
 6:     while $q \neq empty$ do
 7:         $v_0 \leftarrow dequeue(q)$                                  ▷ Get vertex to process
 8:         $activationValue \leftarrow activationFunction(v_0)$           ▷ Calc Act. Value
 9:         if $activationValue \geq threshold$ then               ▷ if the vertex is activated
10:             $setActValue(v_0, activationValue)$
11:             $add(actVertices, v_0)$                       ▷ Collect activated vertices
```

```
12:         for all $v \in adjacent(G', v_0)$ do          ▷ for each adjacent vertex in G'
13:             if $state(v) \neq$ "visited" then
14:                 $setState(v,$ "visited"$)$
15:                 $setEdgeSrcVertex(v, v_0)$      ▷ Save the source vertex of the edge
16:                 $enqueue(q, v)$           ▷ enqueue the adjacent vertix for activation
17:             end if
18:         end for
19:     end if
20:   end while
21:   return $actVertices$
22: end function
                                          ▷ Calc. activation function of the vertex
23: function ActivationFunction($vertex$)
                                          ▷ Get categories of the source vertex
24:   $srcVertex \leftarrow getEdgeSrcVertex(vertex)$
25:   if $srcVertex = null$ then                                    ▷ is a seed?
26:     return 1                                       ▷ return max activation value
27:   end if
28:   $srcCategoryNames \leftarrow getCategoryNames(srcVertex)$   ▷ Array of categories
                                          ▷ Get categories of the current vertex
29:   $vertexCategoryNames \leftarrow getCategoryNames(vertex)$   ▷ Array of categories
30:   $sharedCategoryNames \leftarrow 0$
31:   for all $categoryName \in vertexCategoryNames$ do
32:     if $categoryName \in srcCategoryNames$ then
33:       $sharedCategoryNames \leftarrow sharedCategoryNames + 1$
34:     end if
35:   end for
36:   $actVal \leftarrow \frac{sharedCategoryNames}{length(srcCategoryNames[])} + getActValue(srcVertex) * decayFactor$
37:   return $actVal$
38: end function
                    ▷ Assign weights to every category name of each activated vertex
39: procedure calcWeightsForActVertices($actVertices[]$)
40:   for all $vertex \in actVertices$ do
41:     $categoryNames \leftarrow getCategoryNames(vertex)$
42:     for all $category \in categoryNames$ do
43:       $categoryFreq \leftarrow getClassificationFreq(vertex, category)$
```

```
44:             $weight \leftarrow getActValue(vertex) * categoryFreq$
45:             $setWeight(category, weight)$
46:         end for
47:     end for
48: end procedure
                              ▷ Get a list of terms from categories of activated vertices
49: function GETCATEGORIES($actVertices[]$)
50:     $uniqueCategories \leftarrow Array[]$
                    ▷ Group category names regardless vertices and sum their weights
51:     for all $vertex \in actVertices$ do
52:         $categoryNames \leftarrow getCategoryNames(vertex)$
53:         for all $category \in categoryNames$ do
54:             if $category \notin uniqueCategories$ then
55:                 $newCategory \leftarrow category$
56:                 $setWeight(newCategory, getWeight(category))$
57:                 $add(uniqueCategories, newCategory)$
58:             else
59:                 $existingCategory \leftarrow getCategory(uniqueCategories, category)$
60:                 $weight \leftarrow sum(getWeight(existingCategory), getWeight(category))$
61:                 $setWeight(existingCategory, weight)$
62:                 $update(uniqueCategories, existingCategory)$
63:             end if
64:         end for
65:     end for
              ▷ sort categories in desc. order of weight and discard using the threshold
66:     $getTerms(sort(uniqueCategories, \text{“desc”}), threshold)$
67: end function
                                      ▷ Get relevant terms from the activated vertices
68: function GETRELEVANTTERMS($actVertices[]$)
69:     $calcWeightsForActVertices(actVertices)$              ▷ weights per category
70:     return $getCategories(actVertices)$                      ▷ group categories
71: end function
               ▷ Collect domain relevant terms from the G’ graph using a list of seeds
72: function COLLECTTERMS($G', seeds[]$)
73:     $allActVertices \leftarrow Array[]$
74:     for all $s \in seeds$ do                ▷ For each seed run the spreading activation
```

```
75:         $actVertices = BFS(G', s)$                        ▷ Breath first search over $G'$
76:         $add(allActVertices, actVertices)$                ▷ List of activated vertices
77:     end for
78:     $terms \leftarrow getRelevantTerms(allActVertices)$
79:     return $terms$
80: end function
```

5.2 Semantic elicitation

For the semantic elicitation we propose techniques that rely on knowledge bases published as linked open data (Bizer et al., 2009a). Nowadays linked data is a popular recommendation of best practices to expose, share, and interlink knowledge bases. Publication is done using the resource description framework RDF and uniform resource identifiers URI. Our preference for linked data is based on the fact that all data is exposed using one representation scheme (RDF) which allows querying data using only the SPARQL query language, as opposed to use distinct representation schemes per knowledge base which can lead to the deal with different languages. In addition data sets are interlinked among them allowing benefiting of more than one knowledge base at a time.

We propose, in section 5.2.1, a technique based on the vector space model for the disambiguation of terms when we ground them to semantic entities in the knowledge bases. For the class identification and relation discovery we propose techniques, in section 5.2.2 and 5.2.3, based on dynamic SPARQL queries which are posed on the knowledge bases so that we can obtain the classes and relations among them.

5.2.1 Vector space model for semantic grounding

The semantic grounding for non-ambiguous terms is a direct association between the term and the corresponding entity in the knowledge base. The association is carried out by an exact string matching between the term and the label of the semantic entity. However, for ambiguous terms the semantic grounding has to select from a set of candidate meanings the right one. Hence, we require a knowledge base where the possible meanings of a given term are defined.

We turn the disambiguation problem into an information retrieval (Baeza-Yates and Ribeiro-Neto, 2011) one as follows. First the set of meanings are considered as documents; each document contains the textual description of the corresponding meaning. Then the ambiguous term and its context are used to pose a query over the documents so that the retrieval process must produce the most relevant document based on the overlapping terms between the query and the documents. To implement the retrieval process we use the vector space model (Salton and Mcgill, 1986) which represents as vectors in a multidimensional space documents and queries so that they can be compared using the cosine of the angle formed by the vectors. We define the context of a tag as the set of additional tags co-occurring in a given annotation.

We create a set V as the union of the top N frequent terms in each of the candidate meanings. Next for each candidate meaning we create a vector in $\Re^{|V|}$ where each position corresponds to an element in an ordered version of the V set. The value w_i associated with the i-th position in the vector is calculated using TF-IDF[1] (Baeza-Yates and Ribeiro-Neto, 2011) for the corresponding i-th term in the ordered set. For a document and the i-th term in the vector, TF measures how important is a term in document and corresponds to the frequency of this term in the document. IDF measures how rare in the document collection is a given term. IDF is calculated, (see equation 5.3), as the logarithm of the inverse ratio of candidate documents CD which contain the i-th term with respect to all the candidates. Please note that this approach contrasts with traditional information retrieval since we use only candidate documents for extracting the terms used to create the vectors and for calculating IDF, instead of using the whole set of documents.

$$IDF(t, CD) = log\left(\frac{|CD|}{|\{cd \in CD : t \in cd\}|}\right) \tag{5.3}$$

Similarly, we create a vector for the term and its context. In this case, w_i takes as value 1 if the i-th term in the ordered set appears in the term context, and 0 if not. We compare the term vector and each one of the candidate vectors using as similarity measure the cosine function. Thus, we select the candidate vector with the highest similarity value with respect to the term vector, and return it as the entity to which the term has to be grounded.

Example. Let us suppose we have to ground the term *programming* within a context defined by the terms *instruction*, *order*, and *computer*. The knowledge base has two candidate meanings for programming. The first meaning is *scheduling* which is described as “setting an order and time for planned events”. The second meaning is *computer programming* which is defined as “creating a sequence of instructions to enable the computer to do something”.

Thus, first we create the set V with the terms extracted from the definitions of the

[1]TF-IDF stands for Term Frequency and Inverse Document Frequency

candidate meanings. Each of the terms extracted from the definition were stemmed to get their root form, and then they were filtered using a stopword list. The result set is $V = \{set, order, time, plan, event, create, sequence, instruction, enable, computer, do, something\}$.

Next we represent as vectors in $\Re^{12}$ the candidate meanings; one vector position per each term in V. Note that each term $t \in V$ appears in one of the two meanings, hence $IDF = log(2/1) = 0.3$ for all of them, and the term frequency will be 1 per each document term. We normalize TF by dividing each value by the number of words in each document. Thus for the terms in the "scheduling" meaning $TF = 1/5 = 0.2$ and $TF - IDF = 0.2 * 0.3 = 0.06$. For the terms in the "computer programming" meaning $TF = 1/6 = 0.167$ and $TF - IDF = 0.167 * 0.3 = 0.043$. In table 5.4 we present the vectors created for each of the candidate meanings and for the ambiguous term and its context. Due to space constraints each vector is represented by two rows. In the first row we have vector positions from 1 to 6, and in the second row we have positions from 7 to 12.

Table 5.4: Vector Space for candidates and the query representing the ambiguous term "programming"

scheduling	0.06	0.06	0.06	0.06	0.06	0
	0	0	0	0	0	0
computer prog.	0	0	0	0	0	0.043
	0.043	0.043	0.043	0.043	0.043	0.043
query	0	1	0	0	0	0
	0	1	0	1	0	0

Finally we calculate similarity between the query vector and each one of the candidates: $cos(query, scheduling) = 0.289$ and $cos(query, computerprog) = 0.436$. Give than the cosine of the angle formed by the query vector and the computer programming vector is the highest, we chose this candidate as the most probable meaning of the ambiguous term programming.

5.2.1.1 Algorithm

In algorithm 3 we present the pseudocode for the semantic grounding of terms to semantic entities. The main function is *SemanticGrounding* which receives as input the term to ground as well as the context where the term appears. The context consists of a list of terms. This function retrieves from the knowledge base the set of candidate meanings for the term. In the cases where there are more than one candidate a disambiguation function is used (*disambiguateTerm*). This function creates the vector space of candidates and their terms, as well as a vector for the query which represents the term context. Then, using the *search* function, vectors are compared so that the most similar candidate is retrieved.

Algorithm 3 Semantic grounding of contextualized terms

```
                    ▷ Create the matrix representing the vector space (candidates X allTerms)
 1: function CREATEVECTORSPACE($candidates[], allTerms[]$)
 2:     $vectorSpaceMatrix \leftarrow Matrix[length(candidates)][length(allTerms)]$
 3:     $i \leftarrow 0$
 4:     for all $candidate \in candidates$ do
 5:         $j \leftarrow 0$
 6:         for all $term \in allTerms$ do
 7:             $VectorSpaceMatrix[i][j] \leftarrow calcTFIDF(term, candidate, candidates)$
 8:             $j \leftarrow j + 1$
 9:         end for
10:         $i \leftarrow i + 1$
11:     end for
12:     return $vectorSpaceMatrix$
13: end function
                                        ▷ Create the vector representing the term context
14: function CREATEQUERYVECTOR($context[], allTerms[]$)
15:     $queryVector \leftarrow Array[length(allTerms)]$
16:     $i \leftarrow 0$
17:     for all $term \in allTerms$ do
18:         if $term \in context$ then
19:             $queryVector[i] = 1$
20:         else
21:             $queryVector[i] = 0$
22:         end if
23:         $i \leftarrow i + 1$
24:     end for
25:     return $queryVector$
26: end function
                  ▷ Select from the vector space the most similar candidate to the term context
```

```
function SEARCH(vectorSpaceMatrix[][],queryVector[], candidates[])
    $simCandidates \leftarrow Array[length(candidates)]$
    $i \leftarrow 0$
    for all $candidate \in candidates$ do
        $candidateVector \leftarrow vectorSpaceMatrix[i]$
        $simValue \leftarrow calcCosine(queryVector, candidateVector)$
        $simCandidates[i] \leftarrow candidate$
        $setSimValue(simCandidates[i], simValue)$
        $i \leftarrow i + 1$
    end for
    $sort(simCandidates,$ “desc”$)$                ▷ sort by simValue in descending order
    if $simCandidates[0] \geq threshold$ then
        return $simCandidates[0]$                  ▷ Return the most similar candidate
    else
        return “”
    end if
end function
                 ▷ disambiguate the term searching over the list of candidate meanings
function DISAMBIGUATETERM($termContext[], candidates[]$)
    $allTerms \leftarrow getTerms(candidates)$                ▷ Extract terms of the meanings
    $vectorSpaceMatrix \leftarrow createVectorSpace(candidates, allTerms)$
    $queryVector \leftarrow getQueryVector(context, allTerms)$
    return $search(vectorSpaceMatrix, queryVector, candidates)$
end function
                         ▷ Get the semantic entity for the term meaning in the context
function SEMANTICGROUNDING($term, context[]$)
    $candidates \leftarrow knowledgeBase.getMeanings(term)$          ▷ Array of meanings
    if $length(candidates) = 0$ then
        return “”
    else if $length(candidates) = 1$ then
        return $candidate[0]$
    else
        return $disambiguateTerm(context, candidates)$
    end if
end function
```

5.2.2 Dynamic SPARQL queries for class identification

To identify classes from the semantic entities to which the terms have been grounded, we propose to pose SPARQL queries on the knowledge base so that we figure out whether the semantic entities correspond to ontological classes. Recall that we require that the knowledge base is published following the linked data recommendations, and therefore we can query it with SPARQL.

The most simple case occurs when a term is grounded to a semantic entity which represents a class in the knowledge base. If a semantic entity represents a class then must there exists a RDF statement stating that fact. Therefore we can issue an SPARQL query using the ASK[1] query form to check if the entity is a class (see listing 5.1). This query returns true if the query pattern exists or false otherwise.

```
ASK{<entity> <rdf:type> <rdfs:Class>}
```

Listing 5.1: Query to validate if an entity is a class

Nevertheless, there is the possibility that an entity has not been defined as a class directly but through other linked entities. This case happens, for instance, when a linked data publisher has created a knowledge base KB_i which does not include an ontology, while another publisher has created a knowledge base KB_j enriched with an ontology, and resources of both KB_i and KB_j are linked with *owl:sameAs* relations. Thus, if a term was grounded to a resource in KB_i we ought to follow the link between this resource and the corresponding resource in KB_j to find out if it is defined as a class. In a scenario where we have many knowledge bases we need to be able to follow the links between the resources so that we can benefit the most of the existing information when looking for classes.

Thus, to identify classes related to an entity s we query, using SPARQL, the knowledge bases in order to find paths of *sameAs* relations and of variable length connecting s with a target entity c defined as a class. We follow a similar approach to Heim et al. (2010) where queries are used to traverse all the possible paths in the RDF graph connecting the two entities. We define the path length L as the number of relationships found in the path linking s with c. For $L = 1$ we look for a pattern containing a relationship $relation_i$ linking s with c. As we do not know the direction of $relation_i$, we search in both directions: 1) $s\ relation_i\ c$, and 2) $c\ relation_i\ s$. We present in listing 5.2 the two queries representing the search.

[1] ASK validates the existence of a solution for a query pattern

```
SELECT ?class
WHERE{ <entity> ?rel1 ?class. ?class <rdf:type> <rdfs:Class>
       FILTER (?rel1 = <owl:sameAs>) }
SELECT ?class
WHERE{ ?class ?rel1 <entity>. ?class <rdf:type> <rdfs:Class>
       FILTER (?rel1 = <owl:sameAs>)  }
```

Listing 5.2: SPARQL queries to identify classes (path length 1)

For $L = 2$ we look for a path containing two relationships and an intermediate resource *node* such as: s $relation_i$ $node$, and $node$ $relation_j$ c. Note that each relationship may have two directions and hence the number of possible paths is $2^2 = 4$. Listing 5.3 presents two of the four queries posed for paths of length 2. For $L = 3$ we have two inbetween 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.

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

Listing 5.3: SPARQL queries to identify classes (path length 2)

Thus for each semantic entity e produced in the semantic grounding task and a given value of L we pose n SPARQL queries following the aforementioned pattern to find related classes.

Example. Let us suppose we count on a knowledge base which contains information of three general-purpose linked data sets: DBpedia (Bizer et al., 2009b), OpenCyc[1], and UMBEL[2]. Resources of these data sets are interlinked among them using *owl:sameAs*

[1]OpenCyc home page: http://sw.opencyc.org/
[2]UMBEL home page: http://www.umbel.org/

relations. Our goal is to identify if the DBpedia resource *dbpr:Programmer* corresponds to a class. In figure 5.3 we depict part of the RDF graph where this resource appears. In this graph we can see that *dbpr:Programmer* is not defined as a class in the DBpedia ontology. Hence, if we replace <*entity*> by *dbpr:Programmer* in the query presented in listing 5.1), we obtain *false* as a result.

Thus we need to check if a linked entity leads to a class. for $L = 1$ we pose the 2 queries defined in listing 5.2. The first query finds two solutions matching the query pattern. The first solution states that *dbpr:Programmer* and the class *ocyc:DevelopmentProgram* are the same. The second solution states that *dbpr:Programmer* and the class *ocyc:Developer* are the same. This leads to conclude that *dbpr:Programmer* is actually a Class which is equivalent to the OpenCyc classes *ocyc:DevelopmentProgram* and *ocyc:Developer.*

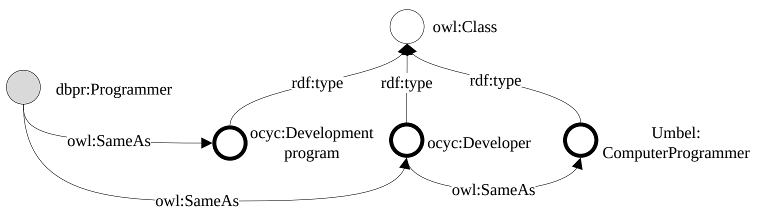


Figure 5.3: RDF graph of data linked to *dbpr:Programmer*. Bold edge nodes represent classes that are reachable, from the initial node which is depicted in gray, following owl:sameAs relationships

If we continue the search for the case where $L = 2$ and pose the corresponding 4 queries we found that the first query presented in listing 5.3 finds a solution. This solution states that *dbpr:Programmer* and *ocyc:Developer* are the same, and that *ocyc:Developer* and the class *umbel:ComputerProgrammer* are the same. Thus we can add to the conclusions the fact that *dbpr:Programmer* is a class equivalent to the UMBEL class *umbel:ComputerProgrammer.*

5.2.2.1 Algorithm

In algorithm 4 we present the pseudocode for identifying relevant classes from the set of semantic entities found in the Semantic Grounding. The main procedure is *IdentifyClasses* which receives as input the semantic entity and a pre-defined path length used to limit the number of relations in the SPARQL queries. *IdentifyClasses* uses a procedure (*generateQueries*) to generate dynamically the SPARQL queries to be posed on the SPARQL endpoint. Then, these queries are executed and the classes are extracted from the result sets.

The *generateQueries* procedure first creates the query (ASK) verifying whether the semantic entity is a class or not. Next it creates the list of queries per each of the possible values of i (1..PathLength). Recall that for each path length value i the number of queries to traverse all the possible paths is 2^i. The strategy to generate the queries per each path length value uses a queue of the resources involved in each query. The queue is created by the function *getResources*, and for each path length value the number of resources in the queue is $pathlength + 1$. For instance for $i = 2$ we have three resources: the semantic entity, an intermediate node (node1), and the class variable.

The creation of each list of queries is delegated to an overload function *createQueries*. This is a recursive function which creates the queries by adding a query pattern at a time using a resource extracted from the queue in each recursive call. For instance, for $i = 2$, in the first call of the *createQueries* function two resources are dequeue and $2^2 = 4$ queries are created with the following query patterns: 1) $r_i \overset{rel1}{\rightarrow} r_j$, 2) $r_i \overset{rel1}{\leftarrow} r_j$, 3) $r_i \overset{rel1}{\rightarrow} r_j$, and 4) $r_i \overset{rel1}{\leftarrow} r_j$. Note that the direction of the relations is alternated according to the *setRelationDirection* function. In a second recursive call, another resource r_k is dequeue, and for each query already created we add new query patterns relating the last node of the existing query pattern to r_k. This results in the following query patterns: 1) $r_i \overset{rel1}{\rightarrow} r_j \overset{rel2}{\rightarrow} r_k$, 2) $r_i \overset{rel1}{\leftarrow} r_j \overset{rel2}{\rightarrow} r_k$, 3) $r_i \overset{rel1}{\rightarrow} r_j \overset{rel2}{\leftarrow} r_k$ and 4) $r_i \overset{rel1}{\leftarrow} r_j \overset{rel2}{\leftarrow} r_k$. Finally the SPARQL query containing these patterns are created by the function *createQuery* for which we do not provide more details since it performs a syntactical transformation between the aforementioned patterns and the SPARQL query syntax.

Algorithm 4 Algorithm for identifying classes using dynamic SPARQL queries

```
        ▷ Give alternate directions for the relation to be included in the query patterns
 1: function SETRELATIONDIRECTION($queryNumber, idRel, currentDirection$)
 2:     if $mod(queryNumber, 2^{idRel-1}) = 0$ then
 3:         if $currentDirection =$ “forward” then
 4:             return “backward”
 5:         else
 6:             return “forward”
 7:         end if
 8:     end if
 9: end function
                        ▷ Create a query containing a pattern for the input resources
10: function CREATEQUERY($resource, nextResource, queryNumber, idRel, direction$)
11:     $direction \leftarrow setRelationDirection(k, idRel, direction)$
```

```
12:     $rel \leftarrow$ "?rel" $+ idRel$                                   ▷ Name of the variable
13:     if $direction =$ "forward" then
14:         $qry \leftarrow createQueryPattern(resource, rel, nextResource)$
15:     else
16:         $qry \leftarrow createQueryPattern(nextResource, rel, Resource)$
17:     end if
18:     return $qry$
19: end function
      ▷ Recursive procedure to create queries for each path length. Queries are created
    by adding relations between pair of nodes in each recursive call.
20: procedure CREATEQUERIES($queries, resourceQueue, currentPathLenght, idRel$)
21:     if $length(resourceQueue) = 0$ then                  ▷ Stop condition of the recursion
22:         return $queries$
23:     end if
24:     $resource \leftarrow dequeue(resourceQueue)$                          ▷ Get a resource
25:     $direction \leftarrow$ "forward"                          ▷ Relation direction: $x \rightarrow y$
26:     if $length(queries) = 0$ then          ▷ Create the initial $2^{currentPathLenght}$ queries
27:         $nextResource \leftarrow dequeue(resourceQueue)$              ▷ Get another resource
28:         for $k = 1 \rightarrow 2^{currentPathLenght}$ do                ▷ For each query to create
                    ▷ Create a query with a pattern relating both resources in a direction
29:             $qry \leftarrow createQuery(resource, nextResource, k, idRel, direction)$
30:             $add(queries, qry)$
31:             $k \leftarrow k + 1$
32:         end for
33:         $idRel \leftarrow idRel + 1$
                              ▷ Recursive call to add another pattern to the queries
34:         $createQueries(queries, resourceQueue, currentPathLenght, idrel)$
35:     else                                    ▷ If the initial queries were already created
36:         $k = 1$
37:         for all $qry \in queries$ do
                    ▷ Query for relation between the last node of qry and the resource
38:             $qryPattern \leftarrow createQuery(lastNode(qry), resource, k, idRel, direction)$
39:             $qry \leftarrow mergeQueries(qry, qryPattern)$
40:             $k \leftarrow k + 1$
41:         end for
42:         $idRel \leftarrow idRel + 1$
```

```
                                        ▷ Recursive call to add another pattern to the queries
        $createQueries(queries, resourceQueue, currentPathLenght, idrel)$
    end if
end procedure
                                        ▷ Create a queue of resources for queries of each path length
function GETRESOURCES($srcResource, targetClass, currentPathLenght$)
    $q \leftarrow queue()$
    $enqueue(q, targetClass)$
    for $j = 1 \rightarrow currentPathLenght - 1$ do
        $node \leftarrow queryPatternVariable($“node” $+ j)$        ▷ node1, node2, etc.
        $enqueue(q, node)$
        $j \leftarrow j + 1$
    end for
    $enqueue(q, srcResource)$
    return $q$
end function
                                        ▷ Start the query generation.
function GENERATEQUERIES($srcResource, targetClass, pathLength$)
    $queries \leftarrow Array[]$
                                        ▷ First query if the srcResource is a class
    $queries[0] \leftarrow$ “ASK{” $+ srcResource +$ “rdf:type rdfs:Class}”
    $relationCounter \leftarrow 1$        ▷ used to name relations: rel1, rel2, etc.
    for $i = 1 \rightarrow pathLength$ do        ▷ For each i value generates all the $2^i$ queries
                                        ▷ Get the (i+1) resources to be used in the queries of length i
        $resourceQueue \leftarrow getResources(srcResource, targetClass, i)$
                ▷ create queries for the current path length (i) using resources in the queue
        $createQueries(queriesPerLength, resourceQueue, i, relationCounter)$
        $add(queries, queriesPerLength)$        ▷ Add the $2^i$ queries to the final lists
        $i \leftarrow i + 1$
    end for
end function
▷ Main procedure: given a semantic entity, and path length defining the number of
links to traverse in the KB, we produce a set of related classes.
procedure IDENTIFYCLASSES($semEntity, pathLength$)
    $classes \leftarrow Array[]$
    $srcResource \leftarrow QueryPatternResource(semEntity)$        ▷ resource <semEntity>
```

```
71:     $targetClass \leftarrow queryPatternVariable($“?class”$)$          ▷ the class variable
72:     $queries \leftarrow generateQueries(srcResource, targetClass, pathLength)$
73:     for all $query \in queries$ do                               ▷ Every query is executed
74:         $result \leftarrow exec($“SPARQLEndPoint”$, query)$
75:         $add(classes, getClasses(result))$                  ▷ get classes from the resultset
76:     end for
77: end procedure
```

5.2.3 Dynamic SPARQL queries for relation discovery

To discover relations between the previously identified classes we also pose SPARQL queries on the knowledge base. We propose to carry out a pairwise search for relationships among the 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 presented in section 5.2.2 for finding classes from the semantic entities. The only difference is that in this case we have a concrete source c_i and a target c_j of the path. Classes found in a path linking c_i and c_j are also considered as classes to the be added to the ontology.

Thus, for a path length $L = 1$ we pose two queries searching for a relationship $relation_m$ linking c_i and c_j. That is we search for the query patterns c_i $relation_m$ c_j and c_j $relation_m$ c_i. Recall that as it was explained in section 5.2.2 the number of queries per each L value is 2^L. For $L = 2$ we pose 4 queries searching for two relations $relation_m$ and $relation_n$ linking an intermediate resource *node* with the two classes. An example of this query pattern is c_i $relation_m$ *node*. *node* $relation_n$ c_j. Intermediate nodes setting a path between the classes are considered as pertinent classes, and thus they can be part of the pairwise search of relationships among the classes. 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.

Example. Figure 5.4 shows RDF graphs depicting existing relationships between the classes *umbel:SoftwareEngineer* and *umbel:ComputerProgrammer*. First we search for the query patterns defined for $L = 1$. We obtain three solutions (see upper left part of the figure) which state: i) that *umbel:SoftwareEngineer* is *subClassOf* *umbel:ComputerProgrammer*, ii) that *umbel:SoftwareEngineer* has as *broaderTransitive* concept to *umbel:ComputerProgrammer*, and iii) that *umbel:ComputerProgrammer* has as *narrowerTransitive* concept to *umbel:SoftwareEngineer*. Next, we search for the query patterns defined for $L = 2$. In this case we get one solution (see upper right part of the figure) stating that *umbel:SoftwareEngineer* and *umbel:ComputerProgrammer* are both *subClassOf umbel:PersonType*. Note that if the class *umbel:PersonType* has not been identified previously we have discovered a new relevant class. Finally, we search for the query patterns defined for $L = 3$ and we get one solution involving two new classes (see bottom part of the figure): *umbel:professional* and *umbel:ComputerProgrammerProfessional.* This solution states that *umbel:SoftwareEngineer* and *umbel:ComputerProgrammerProfessional* are *subClassOf umbel:professional*, and that *umbel:ComputerProgrammer* has as narrowerTransitive concept to *umbel:ComputerProgrammerProfessional.* If we stop here, we collected in total 8 relationships and discovered three new classes.

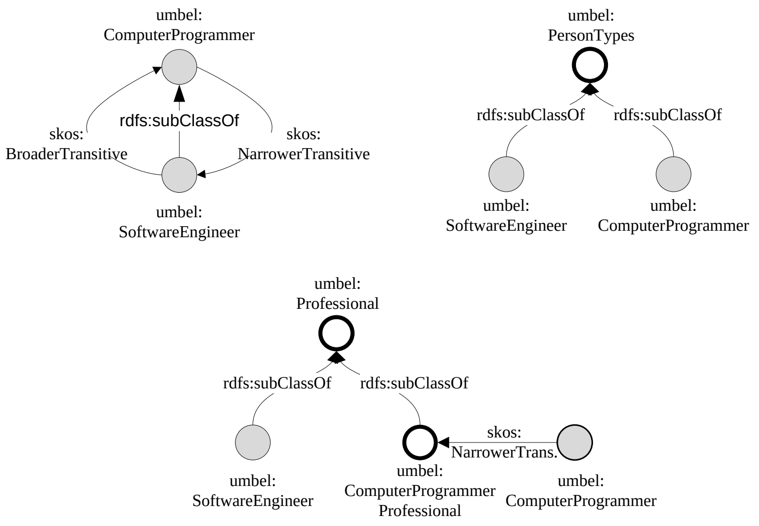


Figure 5.4: Searching for relationships among *ComputerProgrammer* and *SoftwareEngineer*. Bold edge nodes represent new classes discovered in the process.

5.2.3.1 Algorithm

In algorithm 5 we present the pseudocode to discover relations between two classes. This algorithm poses SPARQL queries aiming at traversing all the possible paths (of a predefined length) linking the two input classes inside the knowledge base. The main procedure is *relationDiscovery* which receives as input the two classes and the maximum path length limiting the search of relations linking the two classes. This procedure first generate all the queries, using the function *generateQueries*, which traverse all the possible paths (of length lesser or equal to the predefined length) linking the classes. Next this queries are executed against the knowledge base SPARQL endpoint, and the result of each query is saved as an RDFGraph.

Each RDFGraph is traversed by the *BFS* function, which implements a breath first search, so that we collect the relations discovered in each query. The *BFS* function treats each RDF graph as undirected to reach all the connected nodes, and then obtains each relation, through the *getRelation* function, according to the direction of the edges linking the nodes. From the list of relations we get, using the *getClasses* function, the classes which are different from the input classes and collect them.

For the query generation, in the *generateQueries* function, we use an algorithm similar to the one used in the algorithm for identifying classes. That is we create the queries to traverse all the possible paths linking the two input classes using: 1) a queue of resources to be included in each lists of queries of a given path length, and 2) we use a recursive function which creates the queries for each path length by adding in each recursive call a query pattern involving a resource taken from the queue and a relation. In fact within the code of the *generateQueries* function we reuse the functions *getResources* and *createQueries*. For details of these algorithms as well as an explanation of how they work the reader is referred to Algorithm 4.

Algorithm 5 Algorithm for identifying classes using dynamic SPARQL queries

```
                                                    ▷ Start the query generation.
 1: function GENERATEQUERIES($srcResource, trgResource, pathLength$)
 2:     $queries \leftarrow Array[]$
 3:     $relationCounter \leftarrow 1$                ▷ used to name relations: rel1, rel2, etc.
 4:     for $i = 1 \rightarrow pathLength$ do      ▷ For each i value generates all the $2^i$ queries
            ▷ Get the (i+1) resources to be used in the queries of length i. This function is
            described in algorithm 4 line number 46
 5:         $resourceQueue \leftarrow getResources(srcResource, targetClass, i)$
            ▷ create queries for the current path length (i) using resources in the queue. This
            function is described in 4 line number 20
 6:         $createQueries(queriesPerLength, resourceQueue, i, relationCounter)$
 7:         $add(queries, queriesPerLength)$        ▷ Add the $2^i$ queries to the final lists
 8:         $i \leftarrow i + 1$
 9:     end for
10: end function
                        ▷ create a relation based on the edge direction linking $v_0$ and $v$
11: function GETRELATION($RDFGraph, v_0, v$)
12:     $edge \leftarrow getEdge(RDFGraph, v_0, v)$                                        ▷
13:     $relationName \leftarrow getEdgeName(edge)$
14:     $subject \leftarrow getSourceNode(edge)$
```

```
15:     $object \leftarrow getTargetNode(edge)$
16:     return $setRelation(subject, relationName, object)$
17: end function
                                        ▷ Traverse the rdf graph as an undirected graph
18: function BFS($RDFGraph, srcClass, trgClass$)
19:     $relations \leftarrow Array[]$                              ▷ Array of new relations
20:     $q \leftarrow Queue()$
21:     $enqueue(q, srcClass)$
22:     $setState(srcClass,$ "visited"$)$
23:     while $q \neq empty$ do
24:         $v_0 \leftarrow dequeue(q)$                            ▷ Get vertex to process
25:         for all $v \in adjacent(RDFGraph, v_0)$ do        ▷ for each adjacent vertex
26:             if $state(v) \neq$ "visited" then
                          ▷ Define a relation using the edge direction information
27:                 $add(relations, getRelation(v_0, v))$
28:                 $setState(v,$ "visited"$)$
29:                 $enqueue(q, v)$              ▷ enqueue the adjacent vertix for activation
30:             end if
31:         end for
32:     end while
33:     return $relations$
34: end function
    ▷ Get a list of rdf graphs containing the relationships of variable length found in
  the knowledge base for the two input classes.
35: procedure RELATIONDISCOVERY($srcClass, trgClass, pathLength$)
36:     $allRelations \leftarrow Array[]$
37:     $classes \leftarrow Array[]$
38:     $srcResource \leftarrow QueryPatternResource(srcClass)$
39:     $trgResource \leftarrow queryPatternVariable(trgClass)$
40:     $queries \leftarrow generateQueries(srcResource, trgResource, pathLength)$
41:     for all $query \in queries$ do                      ▷ Every query is executed
42:         $result \leftarrow exec($"SPARQLEndPoint"$, query)$
43:         $rdfGraph \leftarrow getGraph(result)$       ▷ Create an RDF graph from the result
44:         $relations \leftarrow BFS(RDFgraph)$            ▷ BFS to traverse and get relations
45:         $add(allRelations, rdfGraph)$
46:     end for
```

```
            ▷ get the classes from the relations which are different from the input ones
47:     $classes \leftarrow getClasses(allRelations)$
48: end procedure
```

5.3 Conclusions

In this chapter we presented the techniques that we identified and adapted to support the method for developing ontologies from user-generated classification systems. With respect to the Terminology Extraction process we have described techniques for the Normalization task which is part of the Data Preprocessing activity, and for the Term Selection activity. For the Normalization task we propose to use **approximate matching** to match category names to dictionary entries. If the category name matches, or approximately matches a dictionary entry we use the dictionary entry as a normalization of the category name. For the Term Selection we propose to represent the user-generated classification system as a graph and use **spreading activation** to traverse the graph and collect relevant domain category names.

With respect to the Semantic Elicitation process we described techniques for the three activities that comprise this process: Semantic Grounding, Class Identification, and Relation Discovery. For the Semantic Grounding we propose to use the **vector space model** to select, among a set of candidates, the semantic entity that represents a category name. For the Class Identification we propose to use **SPARQL queries** to identify, in a set of knowledge bases, the semantics entities that correspond to classes. Similarly for the Relation Discovery we use **SPARQL queries** to search, in the knowledge bases, for relations between pairs of classes.

CHAPTER 6

EVALUATION

In this chapter we present the different experiments carried out through the development of this thesis which aim at validating in specific scenarios the method proposed to obtain ontologies from user-generated classification systems, thus validating the research hypotheses of this thesis. With the exception of the survey of the emergent semantics in named lists, due to particularities on its design, all the experiment descriptions are presented using a template which consists of the following sections: i) *Data set* where we provide a description of how the data were obtained and statistics about them, ii) *Set up* which is a description of how the evaluation was carried out including the information of the evaluators and of the evaluation metrics, iii) *Evaluation* where the resulting data is tabulated and discussed, and iv) *Conclusions* where we summarize the findings.

In section 6.1 we present an experiment to validate the normalization technique in terms of precision and coverage, which was applied to a set of tags extracted from a popular collaborative tagging site. Next, in section 6.2, we present an experiment about the semantic grounding technique which involved a large number of evaluators and deal with tags in the touristic domain. In addition, we evaluate, in section 6.3, the approach for developing ontologies from user-generated classification systems by generating an initial ontology in the stock market domain from a folksonomy. Finally we have included in section 6.4 the survey of the emergent semantics in user-generated lists created in Twitter. We propose to use existing models to obtain related terms from the classification system, and then we use similarity measures based on WordNet and queries on linked data sets to identify the semantics of the related terms. This type of analysis lays the foundation for the design of procedures to extract knowledge from user-generated lists.

6.1 Evaluation of the normalization

We have carried out an experiment involving real tags, taken from an existing collaborative tagging system, to evaluate, in terms of precision and coverage, the normalization technique proposed in section 5.1.1. First we implement the normalization technique based on the approximate matching of the category names, in this case tags, to dictionary entries. Next, we execute the normalization for a set of tags which have been selected randomly. Finally we manually evaluate the suggestions and present the results in terms of precision and coverage. In the following we present the details of this experiment.

6.1.1 Data set

We used a data set published by Delicious, known as *R5 - Yahoo! Delicious Popular URLs and Tags, version 1.0*, which contains data of 100,000 URLs bookmarked on Delicious at least 100 times per different users. For each URL it is specified the number of bookmarks, as well as the ten most commonly used tags along with the number of times each tag was used to bookmark that resource. In total there are 38,275 tags, and 999,816 assignments of tags to resources.

We selected 5000 distinct tags randomly from the whole set. The random nature of the selection produced a data set where the tag with the minimum number of times used was 1 while the one with the maximum number of times used was 842,972. In average a tag in this data set was used 1,970.149 times with a standard deviation of 21,850.480.

In table 6.1 we show some tags and their frequency of use. To create the table we sorted the tags in descending order of frequency of use and selected the tags according to their position in the sorted list. First column in the table corresponds to the first 10 tags, second column corresponds to the 10 tags in the middle, and the last column to the last 10 tags. Note that most frequent tags are easily recognized as well formed word, while starting with tags in the middle of the list we found tags such as *kansascity* which needs to be splitted to form the real name, while there are some others as *grl* which may be supposed to be the word *girl* though to validate this supposition we ought to use contextual information.

6.1.2 Setup

The normalization technique requires a dictionary where we can look for the entries which correspond to the input tags, and also a popularity index where the dictionary

Table 6.1: Tags according to the frequency of use in the data set

Most used tags		**Tags in the middle**		**Least used tags**	
free	842972	kansascity	50	Bittorent	1
linux	729632	hypnotoad	50	ayhankaya	1
google	485664	plant	50	cuuuuuuuuuuuuuute	1
search	481558	rox	50	importált	1
php	328694	Pan	50	grl	1
science	293302	steorn	50	caute	1
internet	270838	library_of_congress	50	standings	1
wiki	213568	kmz	50	dis	1
mp3	209779	a	50	Faydali	1
research	187303	wmi	50	...	1

entries are stored along with their popularity according to a given corpus. Thus first we create the dictionary as a list of words, using the titles of the articles in Wikipedia as well as the redirection pages. We choose Wikipedia since it is a comprehensive on-line encyclopedia which contains 3,907,396 articles[1]. To create the popularity index we also rely on Wikipedia from which we obtain the number of times that a given word appears in the articles.

To implement the normalization technique we used the lucene framework (see `http://lucene.apache.org/`). With the help of the *SpellChecker* library (see `http://wiki.apache.org/jakarta-lucene/SpellChecker`) we created a lucene index representing the dictionary. The dictionary index contains documents; one document per entry in the dictionary. Each document has as fields the entry as well as its n-grams. Thus within the normalization process this index can be queried to identify entries by comparing the n-grams of both the entry and the query term.

On the other hand, the popularity index contains information of all the articles in the English Wikipedia. There is a document per each article in Wikipedia, and there are two fields, one for the article title, and another for the text in the article. Thus we can query the popularity index by the frequency of appearance of a given word in the Wikipedia articles. In other words, we query the index for the number of documents containing a given word.

In addition we use the Levenshtein distance as similarity measure. This measure was used to calculate the similarity between each tag and its normalized versions. We

[1]data reported in en.wikipedia.org as of March 2012.

set 0.75 as the minimum similarity acceptable between the tag and the normalization.

For each tag in the dataset we queried the dictionary to differentiate the tags that do not need to be normalized, since they already correspond to an entry in the dictionary, from those that need to be normalized. Then we executed the normalization process for the tags that do not match a dictionary entry. Next, we asked an evaluator first to assess the feasibility of evaluating the tag and its normalized version, and second to rate the normalization provided by our approach as valid or not valid. For each tag t_i the evaluator was presented with contextual information consisting of other tags t_j used concurrently when annotating resources in the data set. In addition we asked the evaluator to identify tags written in other language distinct than English.

Metrics To evaluate the performance of the normalization algorithm we use precision and coverage as evaluation metrics. The precision is measured as the fraction of normalizations that are valid, while coverage is the fraction of tags for which we obtain a normalization.

6.1.3 Evaluation and discussion

In table 6.2 we present the results of this evaluation. From the 5000 thousand tags we found that 1660 (33.2%) required preprocessing to match a dictionary entry. Our normalization algorithm found an alternative spelling in the dictionary for 1259 of them, and hence producing a normalized version for 75.843% of the tags. The evaluator was unable to evaluate 386 (30.659%) of the tags due to the impossibility of understanding the tag meaning. The evaluator rated, out of the 873 evaluated, 628 normalizations as valid which leads to a precision of 71.936% and a coverage of 75.843%.

Table 6.2: Evaluation of the normalization of tags using approximate matching

Statistics	All tags	Tags in English
Processed Tag	1660	1462
Normalized Tags	1259	1061
Precision	71.936%	73.315%
Coverage	75.843%	72.571%
Non Evaluated	30.659%	32.893%

Note that since we found that 11.928% of the tags were written in a language different than English we include this diferentation in the reported data. For tags only in English the precision was improved in 1.379% while the coverage was descreased in -3.272%. A possible reason explaining why precision increased is the fact that some words in other languages have a similar spelling to words in English but their meaning is different leading to a non valid normalization. This is the case of words such as *Precios* in spanish which means prizes and *Precious* in english. On the other hand, the coverage decreased since when filtering out non-English tags we are decreasing the overall number of valid relations while keeping constant the number of tags processed. Though the coverage in both cases may seems low we consider that it is a satisfactory value since even for the evaluator was difficult to asses the meaning of around 30% of the tags.

Finally in table 6.3 we present some of the tag normalizations. First column shows valid normalizations where we can see that our technique is able to: 1) normalize plurals to singular forms (*e.g.*, *adjustments*), 2) split composed words (*e.g.*, *Accountplanning* and *Adamandjoe*), 3) obtain words from composed strings (e.g., *101cookboks* and *6degrees*), and 4) approximate words in other languages which are very similar to the corresponding words in English (*e.g.*, *directorios* in Spanish).

Table 6.3: Examples of tag normalizations. Tags are presented with their corresponding normalized version.

Valid Normalizations		**Wrong Normalizations**		**Non-Normalized**
Adjustments	Adjustment	Activemailer	Active_filter	200dinge
Accountplanning	Account_planning	Bluemark	Blueback	21stcenturyskills
Adamandjoe	Adam_and_joe	Eatability	Castability	Appleespaña
Direct-action	Direct_action	Gifs	Gifts	Firefox:toolbar
Bandwidth-speed	Bandwidth_speed	Group48	Group_8	360doc
Cdcovers	Covers	Hyperfocal	Hyperlocal	3gb
101cookbooks	Cookbooks	Jee	Jeep	3voor12
6degrees	Degrees	Owasp	Wasp	Alink=#ff0000""
Directorios	Directories	Carburante	Carborane	Bancodedados

Second column shows wrong normalizations which occur mainly because: 1) there are not corresponding entries in the dictionary for the concepts involved in the tags (e.g., *Activemailer* and *bluemark*), 2) some of the tags do not represent an accepted concept (e.g., *eatability*), 3) the tag was wrongly matched with an existing concept (e.g., *Jee* refers to J2EE but was normalized to Jeep), and 4) words in other languages were wrongly matched with similar English words (*e.g*, *Carburante* which means fuel in Spanish).

Finally in the third column we present some tags for which the normalization procedure did not find an entry in the dictionary. In this case we see a variety of tags including: 1) tags mixing words in two different languages (*e.g.*, *Appleespaña*), 2) tags representing composed words which are related but do not form a concept (e.g., firefox:toolbar), 3) tags mixing numbers and non-alphabetical characters (e.g., *Alink=#ff0000""*), and 4) tags in other languages which are not similar to a word in English (*e.g.*, *bancodedados* which means data base in Portuguese)

6.1.4 Conclusions

We have presented an experiment to evaluate the use of approximate matching to normalize tags. In this experiment we found that 33.2% of the tags were not found in the dictionary. This fact confirms the need of a preprocessing activity to deal with this percentage of tags so that some of them can be normalized according to dictionary entries. We also showed that not all the tags can be normalized since even humans were not able to identify the intended meaning of 30.659% of them. In addition we can report, based on this experiment, that this technique is able to produce normalizations with precision of 71.936% and coverage of 75.843%.

Since the Normalization task is part of the Data Preprocesing activity of the Terminology Extraction process (see sections 4.2 and 5.1)this experiment is directly related to the objectives **O1** and **O3**, and contributions **C1** and **C2** (see chapter 3).

6.2 Evaluation of the semantic grounding

To evaluate the semantic grounding, in terms of precision and recall, we experimented with tags of a photo sharing site. The set of tags was multilingual and therefore we adapted the grounding process to deal with multiple languages. We use DBpedia as a knowledge base where the semantic entities can be considered as concepts which can be used in the grounding process. We consider different decisions that can be taken in the process with regard mainly to the source of terms representing each concept in the vector space model. We run the semantic grounding for each tag in the data set and ask a group of evaluators to rate manually the results. We use as evaluation metrics *Precision, Recall, Mean Average Precision MAP*, and *F-Measure* on which we base the discussion of the results. We validated the ratings generated by the evaluators using standard measures of agreement as Fleiss' kappa statistic.

6.2.1 Data set

We used as test data a set of tagging activities taken from Flickr[1]. By exploring Flickr images we found that some pictures of tourist cities were annotated with multilingual tags. Thus, we queried the Flickr API[2] for pictures tagged with touristic places in Spain (e.g., Barcelona, Canary Island, Ibiza, etc.). We gathered a total of 764 photos uploaded to Flickr by 719 distinct users. On average those 764 photos were annotated using 12.4 tags with a standard deviation of 7.85. In addition, our data set consists of 9484 tagging activities, that is, 9484 triples of the form $\langle user, tag, photo \rangle$, where 4153 distinct tags were used. Each tag was used on average 2.28 times to annotate the pictures with a standard deviation of 5.69.

6.2.2 Setup

The evaluation focused on determining the precision of the semantic grounding, considering different decisions that can be taken in the process regarding the context of ambiguous tags and the words used to represent each sense (*i.e.*, semantic resource) in the vector space model. In case of context we were interested in defining whether we should use all the tags co-occurring with the ambiguous tag in the annotation activity in contrast to a more reduced set of related tags. In case of sense representation we wanted to evaluate whether we should use all the words appearing in the description of the resource or a smaller set of words taken from its definition.

Active Context Selection Traditional disambiguation techniques utilize many context features for disambiguation (part-of-speech, collocation, surrounding words, etc.) when dealing with ambiguities in well-formed texts and sentences. Unfortunately, these features are not available in certain Web-based systems, where the context consists of unstructured bags of words, such tags in folksonomies. We are interested in the tag meaning in each annotation, and hence we define the context of a tag as the set of additional tags co-occurring in a given annotation. However, many tags refer to subjective impressions of users (e.g., *my favourite*, *amazing*) or technical details (e.g., *Nikon*, *photo*) which can be useless (or even harmful) for disambiguation. Therefore, we need to select among all tags in the context those which help most to figure out the tag meaning.

To carry out this selection we use a technique described in (Gracia and Mena, 2009) which relies on the following hypothesis: the most suitable context words for

[1]flickr home page `http://www.flickr.com/`

[2]The API is available here `http://www.flickr.com/services/api/`

disambiguation are the ones most highly semantically related to the ambiguous keyword. Based on that, we use a simple mechanism to select the active context: After removing repeated words and stop words from the context, we compute the semantic relatedness between each context word and the word to disambiguate. This relatedness computation is performed by using a web-based relatedness measure, similar to the Normalized Google Distance (Cilibrasi and Vitanyi, 2004), which takes into account the co-occurrence of words on web pages, according to frequency counts, and giving a value between 0 and 1, which indicates the degree of semantic relatedness that holds between the compared words. Finally, we construct the active context set with the context words whose relatedness score above a certain threshold.

Representing Semantic Resources We were interested in evaluating how well the semantic grounding performs when the keywords representing each sense in the vector space model are the most frequent terms in the content of the Wikipedia articles related to each DBpedia resource, in contrast to a more reduced set of terms extracted from article abstracts (i.e, the first paragraph describing the article content).

We also consider a baseline which attempts to directly relate tags to DBpedia resource names using exact string matching. From now on we use the term *SemGro* to refer to the semantic grounding. In our experiment we evaluated the following approaches for the semantic grounding of tags:

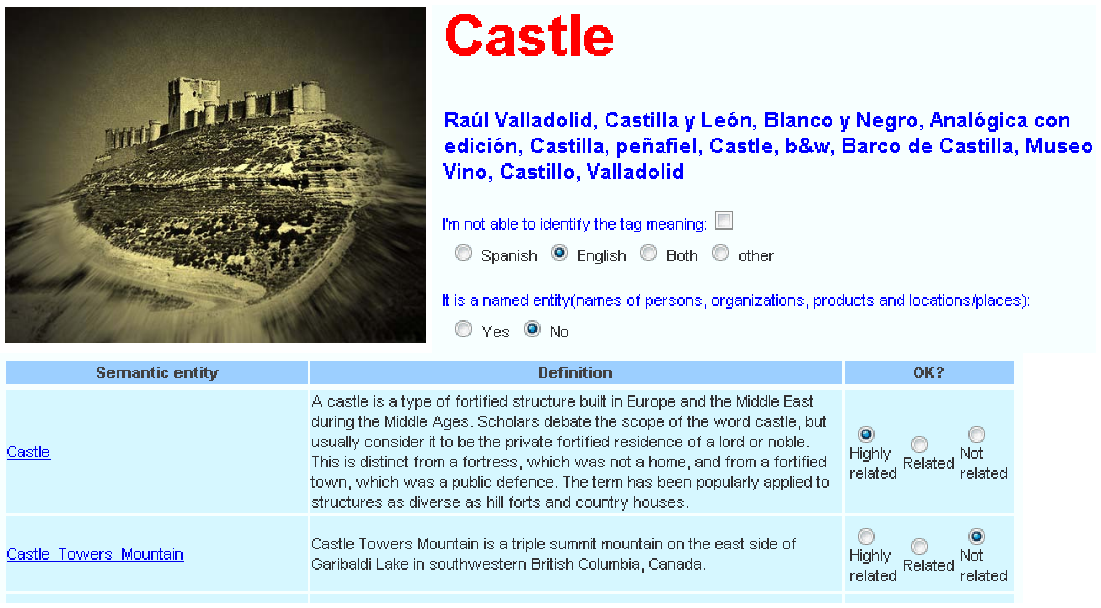
- **Baseline**: Selection of the sense without disambiguation nor preprocessing.
- **SemGro**: For each sense we use the whole Wikipedia article as source for frequent terms.
- **SemGroAC**: Same as SemGro including the selection of the Active Context.
- **SemGroAbs**: For each sense we use the first paragraph of the Wikipedia article as source for frequent terms.
- **SemGroAbsAC**: Same as SemGroAbs including the selection of the Active Context.

We engaged 41 evaluators in the evaluation campaign. Each of them had to evaluate a set of semantic associations[1] generated by each approach, and for that evaluators were

[1]Tuple of the form $\langle user, tag, photo, DBpedia\_resource, language\rangle$

presented with the top 5 semantic entities produced by each approach[1]. We made sure that each semantic association was evaluated by at least three evaluators so that we can use those decisions taken by user majority.

For each tagging activity evaluators decided whether they were able to identify the semantics of the tag. Then they had to identify the language of the tagging activity so that they evaluate the semantics associations accordingly. We presented to them the set of DBpedia resources (title and abstract) returned by all the approaches in the first 5 positions. They were asked to state if each DBpedia resource associated with the tagging activity was highly related (HR), related (R), or not related (N). Note that the evaluation was blind since evaluators did not know where the semantic entities were coming from (*i.e*, the approach), neither the semantic entities were presented in a predefined order. A screenshot of the evaluation application is shown in figure 6.1. In this application the tag to ground is at the top, and the context tags are below in bold.



Castle

Raúl Valladolid, Castilla y León, Blanco y Negro, Analógica con edición, Castilla, peñafiel, Castle, b&w, Barco de Castilla, Museo Vino, Castillo, Valladolid

I'm not able to identify the tag meaning: ☐

Spanish [✓ English] Both other

It is a named entity(names of persons, organizations, products and locations/places):

Yes [✓ No]

Semantic entity	Definition	OK?
Castle	A castle is a type of fortified structure built in Europe and the Middle East during the Middle Ages. Scholars debate the scope of the word castle, but usually consider it to be the private fortified residence of a lord or noble. This is distinct from a fortress, which was not a home, and from a fortified town, which was a public defence. The term has been popularly applied to structures as diverse as hill forts and country houses.	[✓ Highly related] Related Not related
Castle_Towers_Mountain	Castle Towers Mountain is a triple summit mountain on the east side of Garibaldi Lake in southwestern British Columbia, Canada.	Highly related Related [✓ Not related]

Figure 6.1: Screenshot of the interface where users evaluated the grounding of tags

Metrics We use precision and recall as evaluation metrics. For a given approach and tag, precision is defined as the fraction of DBpedia resources retrieved by the approach that are actually related to the tag. Since our final goal is to obtain a single related resource, we measure average precision values taking into account only the first results

[1]Note that evaluators did not know where semantic associations were coming from

returned by each approach (*i.e.*, precision at one or $P@1$). For more exhaustive comparisons, we also compute $P@N$, with $N = 2, 3, 4, 5$. Furthermore, averaging the sum of $P@N$ values by the number of related resources, we define the mean average precision MAP. In turn, recall is defined as the fraction of DBpedia resources related to the tag that are successfully retrieved by the approach. Similarly to precision, we also take into consideration recall at N or $R@N$, with $N = 1, 2, 3, 4, 5$. Finally we use the well known F measure, which is the weighted harmonic mean of precision and recall: $F = 2 \cdot precision \cdot recall/(precision + recall)$.

6.2.3 Evaluation and discussion

We evaluated a total of 2260 tagging activities (TAS) corresponding to 764 pictures tagged with 1112 tags[1]. Evaluators were able to identify the semantics of 87% of the TAS. From this subset, 62.6% were considered in English and 87.7% in Spanish. Statistics about the evaluation are presented table 6.4.

Table 6.4: Description of the evaluated tag assignments.

	Users	Evaluations	Evaluations/ user	Pictures	Tags	TAS	TAS/ picture
English tags	41	30400	741.46 (±206.51)	642	659	1232	1.92 (±0.79)
Spanish tags	41	49568	1208.98 (±152.10)	742	816	1727	2.33 (±0.74)

From the set of tags that evaluators were able to identify their meaning and language, our process associated the 86.9% of tags in English, and 86.7% in Spanish to DBpedia resources. The preprocessing activity was useful to find a DBpedia resource name for the 76.4% of the tags in English and 76.6% in Spanish.

Precision and Recall Analysis Table 6.5 shows the results obtained by the different approaches on tags marked as English and Spanish respectively. For a given tag, based on the three users' evaluations, a semantic entity was considered relevant if at least two users stated it was *highly related* (or *related/highly related*) to the tag. There was a substantial agreement among users. Fleiss' kappa statistic Fleiss and Cohen (1973) measuring users' agreement was $\kappa = 0.76$ (a value $\kappa = 1$ means complete agreement) for the *highly related* case, and $\kappa = 0.71$ for the *related/highly related* case. In the reported results, the former case was used because of its higher agreement level. Similar average performance results were obtained with the latter case. Precision values were higher and

[1]Dataset available in `www.oeg-upm.net/index.php/en/material-used-papers`

recall values were lower. There were more relevant entities so it was easier to accurately retrieve a relevant entity, while it was more difficult to retrieve all relevant entities. We also measured the agreement when identifying the language. There was an almost perfect agreement among users. Fleiss' kappa statistic was $\kappa = 0.83$.

Table 6.5: Evaluation results achieved by the different approaches.

	MAP	P@1	P@2	P@3	P@4	P@5	R@1	R@2	R@3	R@4	R@5	F
English tags												
Baseline	0.78	0.88	-	-	-	-	0.78	-	-	-	-	0.28
SemGro	$\underline{\mathbf{0.91}}$	0.89	0.53	0.37	0.29	0.23	0.81	0.91	0.93	0.95	0.96	$\underline{\mathbf{0.36}}$
SemGroAC	$\underline{\mathbf{0.90}}$	0.90	0.52	0.36	0.28	0.23	***0.82****	0.90	0.92	0.93	0.94	$\underline{\mathbf{0.36}}$
SemGroAbs	***0.84***†	0.82*	0.48	0.34	0.26	0.22	0.75*	0.85	0.89	0.90	0.92	**0.34**
SemGroAbsAC	***0.86***†	0.86	0.48	0.34	0.26	0.22	0.79	0.86	0.89	0.90	0.92	**0.34**
Spanish tags												
Baseline	0.71	0.88	-	-	-	-	0.71	-	-	-	-	0.27
SemGro	$\underline{\mathbf{0.93}}$	**0.93**	0.58	0.42	0.33	0.27	**0.79**	0.90	0.95	0.97	0.98	$\underline{\mathbf{0.41}}$
SemGroAC	$\underline{\mathbf{0.93}}$	**0.94**	0.57	0.42	0.33	0.27	**0.80***	0.89	0.93	0.96	0.96	$\underline{\mathbf{0.40}}$
SemGroAbs	$\underline{\mathbf{0.88}}^{\ddagger}$	0.90*	0.53	0.39	0.32	0.26	***0.76****	0.85	0.90	0.93	0.94	$\underline{\mathbf{0.39}}$
SemGroAbsAC	$\underline{\mathbf{0.89}}^{\ddagger}$	0.91*	0.54	0.40	0.32	0.26	***0.77***	0.85	0.90	0.93	0.94	$\underline{\mathbf{0.39}}$

The results shown in the table were obtained from those tagging activities where the associated semantic entities were known for the evaluators, and in which the corresponding tags were linked to DBpedia resources by at least one approach. Note that *recall* is computed assuming that the set of *all* tags relevant to a given tag is composed by the *relevant* (see definition above) entities retrieved by the investigated approaches. We cannot assure that we are able to retrieve all relevant entities but a strong representative sample of them.

Wilcoxon's statistical tests were performed for $MAP, P@1, R@1$ and, F-measure to determine whether there were statistical significance differences between the metric values obtained with the baseline and the proposed approaches, and between the metric values obtained with SemGro approach and its variants. The statistical tests were applied on those tagging activities where all approaches (including the baseline) were able to link at least one DBpedia resource. This allows us to present a more fair comparison among approaches, but implies a loss of information that hides a higher statistical evidence in the differences with metric values of approaches able to link DBpedia resources in a large number of cases. In table 6.5 values in underline bold (p=0.01), bold (p=0.05), and italic bold (p=0.1) indicate a statistical significance difference with values achieved by the baseline approach. Values marked with ‡(p=0.01), †(p=0.05), and *(p=0.1) indicate a statistical significance difference with values achieved by SemGro approach.

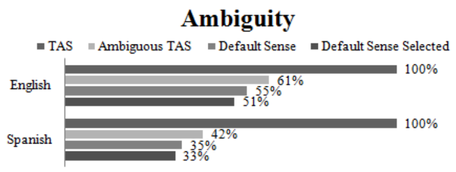


Figure 6.2: Ambiguity of tags with relevant results produced by the *SemGro* approach

Finally, since the baseline retrieves a single semantic association for each tag, the metrics $P@N$ and $R@N$ with $N = 2, 3, 4, 5$ are not reported for that approach. Indeed, the coverage (recall) of the baseline is low in comparison to the proposed approaches, as shown in the tables. The following conclusions can be drawn from our study:

- In general, the baseline had a good performance with tags in English and in Spanish. This fact suggests that a high percentage of the analyzed tags were used in the sense directly found by the Baseline which corresponds to the Wikipedia default sense (*i.e.*, those wikipedia pages that editors have defined to display first in case of ambiguous terms). Its high P@1 value is due to the fact that in the 90% of TAS in English and 91% in Spanish the correct sense corresponds with the default sense. Nevertheless, the coverage of the baseline, defined as the number of semantic associations produced by the baseline divided by the total number of TAS, is extremely low: 27.7% in English and 19.4% in Spanish. This contrast, with the 79.1% of SemGro coverage in English and 81.4% in Spanish. This difference in coverage is due to the preprocessing activity.

- SemGro and its variants perform better when dealing with Spanish tags. The amount of information in the Spanish Wikipedia compared with the English version is considerably lower[1]. Less articles in the Spanish version may indicate less ambiguity in the sense that not all the possible meanings of a word have been added to the Wikipedia. In fact, the average of senses was 23.3 for English and 10.35 for Spanish. As shown in figure 6.2 there were less tags in Spanish considered ambiguous (42%) than in English (61%), and thus the grounding was straightforward for more tags in Spanish (58% of non ambiguous tags) than for tags in english (39% of non ambiguous tags).

[1]As of April 2012, the English and Spanish Wikipedia have 3,921,259 and 882,859 articles respectively

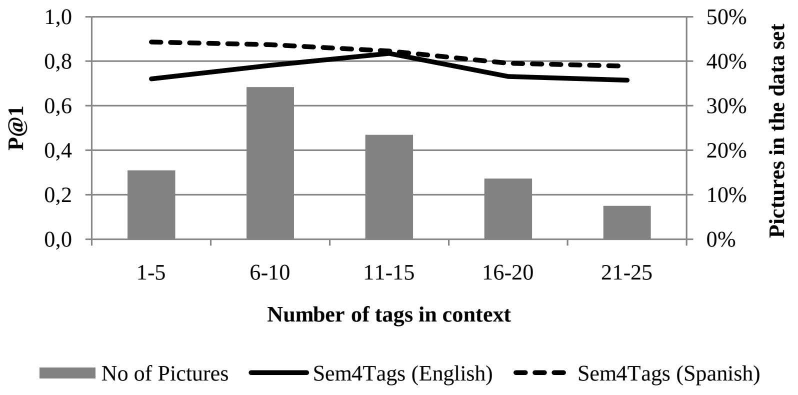


Figure 6.3: Precision variation according to the context length

- SemGro and SemGroAC were the approaches that obtained the best results both in term of precision and recall. Almost all of these results present statistical significant differences with results obtained with the baseline. Comparing SemGro and SemGroAC, we do not find a clear enhancement of semantic associations when exploiting the active context. In some cases, it seems that SemGroAC obtains better P@1 and R@1 values, but the improvements are supported by no or low statistically evidence. This observation could be biased by the way in which statistical tests were conducted, as explained before.
- SemGroAbs and SemGroAbsAC are the worst approaches. Abstract terms do not provide enough information to properly disambiguate the tag meaning. That is, the scarcity of terms in the abstract decreases the overlapping of these terms with tags in the context.
- In the 17% and 20% of ambiguous tags in English and in Spanish respectively, the correct sense was different from the Wikipedia default sense. Therefore validating the need of a disambiguation activity.
- While SemGro precision is related to the number of tags in the context, it presents different patterns for tags in English and Spanish (see figure 6.3). In the case of tags in English, pictures annotated with between 6 and 15 tags (representing 58% of the total) produce the highest P@1 reaching a peak for context containing between 11 and 15 tags. Short contexts with less than 5 tags, or long contexts with more than 16 tags produce, though satisfactory, lower P@1 values around 71%. Short contexts do not provide enough evidence (*i.e.*, words to measure the overlapping with words in the candidate senses) to select the right sense for a tag.

In contrast, long contexts are noisy in the sense that some words in the context can indicate one possible meaning while other words point to other meaning. In case of tags in Spanish, short contexts does not affect Sem4Tag precision. In fact, the highest precision is achieved when context length range between 0 an 10. Nevertheless, starting from 15 precision decreases along with the context length until it stabilizes around 78%.

The exploitation of the active context of a tag seems to improve the performance of our approach. Nonetheless, we do not obtain statistically significant evidence to support that claim. Additional evaluations focused on measuring the importance of semantic context have to be done.

6.2.4 Conclusions

We have presented an experiment where we evaluate the performance, using precision and recall, of the semantic grounding of category names, in this case tags, to semantic resources in the DBpedia knowledge base. We evaluated different versions of this process which were defined according to possible variations of the context definition and the sense representation in the vector space model. We also define a baseline where tags where grounded to semantic entities with labels equals to the them.

We found that the semantic grounding process performs better when the senses were represented in the vector space model by all the words found in the Wikipedia articles which describe the corresponding senses (*i.e.*, the semantic resources in DBpedia). On the other hand, with respect to the context definition we found that the two version of the context produce similar results. In fact with the collected data we could not claim any statistical difference between the results using both context definitions. We also found that for 90% of the tags in english the correct sense corresponds to the default sense defined in the Wikipedia for the terms, and hence the baseline achieved a satisfactory precision of 88%. However for the 17% of ambiguous tags in English, which were 61% of the total, the correct sense was different from the default sense and this justify the need of disambiguating the meaning of the tags.

Since the Semantic Grounding activity is part of the Semantic Elicitation process (see section 4.3.1 and 5.2.1) this experiment is directly related to the objectives **O1**, **O2** and **O3**, and contributions **C1** and **C2** (see chapter 3).

6.3 Evaluation of the method for developing ontologies

To evaluate the method for obtaining ontologies from user-generated classification systems we carried out an experiment which involves the generation of a preliminary ontology in the stock market domain from a folksonomy excerpt extracted from Delicious. We used general purpose knowledge bases which are published as linked open data. In addition, we used relevant financial resources provided by domain experts as seeds during the terminology extraction process. We evaluated the results of the two main processes of the method using, in case of the terminology extraction, an automatic procedure to compare the output terminology to existing knowledge resources in the domain, and in case of the semantic elicitation we analyzed the output ontology with the help of automatic tools as well as with some experts who rated its relevance to the domain.

6.3.1 Data set

As source of folksonomies we used "Delicious Popular URLs and Tags, version 1.0" which was introduced previously in section 6.1.1. In short this dataset was comprised of 100,000 URLs; each URL had been saved at least 100 times. The ten most commonly used tags for each URL as well as the number of times each tag was used were available in our initial dataset.

6.3.2 Setup

Terminology extraction Recall that spreading activation was the technique defined to be used in the term selection activity which is part of the terminology extraction process. Spreading activation requires a list of seed nodes from which the traverse of the graph can start. In this case the seeds must be URL of resources bookmarked in Delicious. Therefore we asked some domain experts to indicate domain prominent web pages in the stock market domain which are part of the Delicious dataset. The suggested seeds are presented in table table 6.6.

Since the terminology gathered by the spreading activation technique 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 the activated resources.

For the purpose of evaluating the relevance of the extracted terms, we compared our

Website	Description
www.google.com/finance	Information including markets, news, currency, etc.
money.cnn.com	Fortune and Money financial magazines
www.ft.com	International business newspaper
www.nasdaq.com	NASDAQ Stock Market
www.federalreserve.gov	Federal Reserve, the central bank of the United States
www.marketwatch.com	Information of business news, analysis and stock market
www.nyse.com	New York Stock Exchange market
www.ecb.int	European Central Bank
londonstockexchange.com	London Stock Exchange market
finance.yahoo.com	Information that includes news, international market data and stock quotes, etc.

Table 6.6: Financial-related websites used as seeds of the term selection activity.

domain terminology to three financial glossaries: Yahoo! Financial Glossary[1], Investor Words [2], and Campbell R. Harvey's Hypertextual Finance Glossary[3].

Semantic elicitation For the semantic elicitation process we used as knowledge bases DBpedia (Bizer et al., 2009b), OpenCyc[4], UMBEL[5] and YAGO (Suchanek et al., 2008). All these knowledge bases are general purpose in the sense that they cover not a unique domain but several, and are published as linked data. DBpedia 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. YAGO is a knowledge base derived from Wikipedia and WordNet; its core version has over 2.6 million entities and around 33 million facts. These datasets are interlinked among them. DBpedia resources, and classes are connected to OpenCyc concepts using *owl:sameAs*, to UMBEL concepts using *umbel#correspondsTo*, and to YAGO concepts using *rdf:type* and *owl:sameAs*.

The semantic elicitation activities tap into different knowledge bases to carry out their functions. For instance the semantic grounding uses DBpedia resource to ground

[1]see `http://biz.yahoo.com/f/g/`

[2]see `http://www.investorwords.com/`

[3]see `http://www.duke.edu/$\sim$charvey/`

[4]OpenCyc home page: http://sw.opencyc.org/

[5]UMBEL home page: http://www.umbel.org/

the terminology, leveraging the high degree of interconnection offered by this knowledge base with respect to the linked data sources. On the other hand, the class identification activity harnesses the DBpedia ontology and interconnected data sets together with their corresponding ontologies. Similarly, the relation discovery activity uses all the linked data sets to look for relationships among the previously identified classes.

To evaluate the output ontology we asked to 4 domain experts to assess the relevance of the classes and relations in the stock market domain. We asked the evaluators to rate classes in each module according to the relatedness of each class to the financial domain. They were asked to state whether a class was *highly related*, *related*, or *not related* following some guidelines that we provided. After collecting the results we measured the precision of the ontology classes over those decisions for which the majority of evaluators reached an agreement. In other words, at least three evaluators agreed with the rating of the class. A class was considered relevant if at least three evaluators asserted that it was highly related or related.

To evaluate the relationships between classes we express them in natural language sentences using the SWAT tool [1] so that evaluators were able to state the truth value of them. SWAT receives as input an ontology and produces as output a set of sentences in natural language describing the ontology axioms. For instance, if we have an ontology asserting that Stock *rdfs:subClassOf* Equity and *owl:disjointWith* LoanNote, the SWAT tool 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.

We also asked evaluators to validate the existence of each of classes of the output ontology in the SUMO financial ontology. The Suggested Upper Merged Ontology[2] (SUMO) is a general ontology which contains knowledge of different domains including the financial. SUMO finance is a broad ontology including personal banking and stock market conceptualizations. Thus, we selected the classes that were related to the stock market sub-domain.

Metrics

To evaluate the **Terminology extraction** process we compare the extracted terms with existing terms in glossaries in the financial domain. We calculated precision as the fraction of relevant terms with respect to all the terms found by the terminology extraction process. A term was considered relevant if it exists in the corresponding glossary.

[1]`http://swat.open.ac.uk/tools/`

[2]`http://www.ontologyportal.org/`

For the **Semantic Elicitation** process we evaluate the output ontology in terms of precision, and recall. We calculate precision to evaluate the identified classes and the discovered relations. For the former case precision was defined as the fraction of relevant classes identified by the domain experts in the ontology. For the latter case precision was calculated as the fraction of valid relations identified by the domain experts from the natural language sentences which describe the axioms in the ontology.

On the other hand, recall was calculated for the ontology classes with respect to existing classes in the stock market sub-domain of the SUMO finance ontology. Thus, we measured recall as the fraction of the SUMO classes in the stock market sub-domain which were equivalent or subsumed by one class of the output ontology.

6.3.3 Evaluation and discussion

Terminology extraction. The precision of the Terminology Extraction process is reported in Table 6.7. The different lists of terms produced by the spreading activation when varying the values of the threshold h were compared to the terms in the financial glossaries. Using a threshold of 0.8 we got a terminology with the highest average precision. We compared the 58 terms against the Yahoo! glossary 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,' 'Depository Institutions Deregulation' and 'Monetary Control Act.' Such overly specialized terminology was not found to be common in folksonomies.

		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**%

Table 6.7: Precision of the terminology gathered by our method using different threshold values.

Semantic elicitation. We executed the semantic elicitation process for the 58 terms gathered in the previous phase using a threshold of 0.8. Results are shown in Table 6.8. 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 grounded terms and that of DBpedia resources to which they were grounded exposes that some of the 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.

Semantic Elicitation				**Output Ontology**					
Grounding Terms	**Identifying Classes**	**Finding Relationships**		**Classes**	**Relations**	**Relations Types**	**Links**		
42 dbpr	23 classes	378 relations	164 new classes	187	378	8	184 ocyc	26 umbel	2 dbpo

Table 6.8: Statistics of the semantic elicitation process and of the output ontology

We also found 378 relationships and 164 new classes. The final ontology[1] 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: *rdfs:subclassOf, owl:disjointWith, owl:sameAs, ocyc:SiblingDisjointExceptions, ocyc:RewriteOf. ocyc:Facets-Partition, ocyc:TypeGenls, ocyc:Facets-Generic*[2]. Of the 187 classes, only ten were totally disconnected from the others. An excerpt of the produced ontology is shown in Figure 6.4.

In order to analyze the knowledge within the generated ontology, we decomposed it into modules using the partitioning tool PATO (Schlicht and Stuckenschmidt, 2008). 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; the isolated nodes are also identified. Modularity of an ontology can be considered as a quality indicator since modules facilitates ontology analysis and maintenance as well as they make easier to avoid inconsistencies and to identify semantic defects (Schlicht and Stuckenschmidt, 2008).

For our resulting ontology, PATO identified eleven modules of interconnected nodes and one module of isolated nodes; the interconnected modules are presented in figure 6.5. From these modules we observed that our ontology was composed of a majority of topics that are relevant to the financial domain. For instance, there are modules describing: 1) financial tasks such as Stock Exchange and Money Transactions, and 2) agents participating in those activities, including Organizations, Companies, Bankers,

[1]The financial ontology can be downloaded from http://delicias.dia.fi.upm.es/wiki/ images/3/34/FinanceOnto.zip

[2]For more information regarding openCyc relationships refer to the ontology documentation.

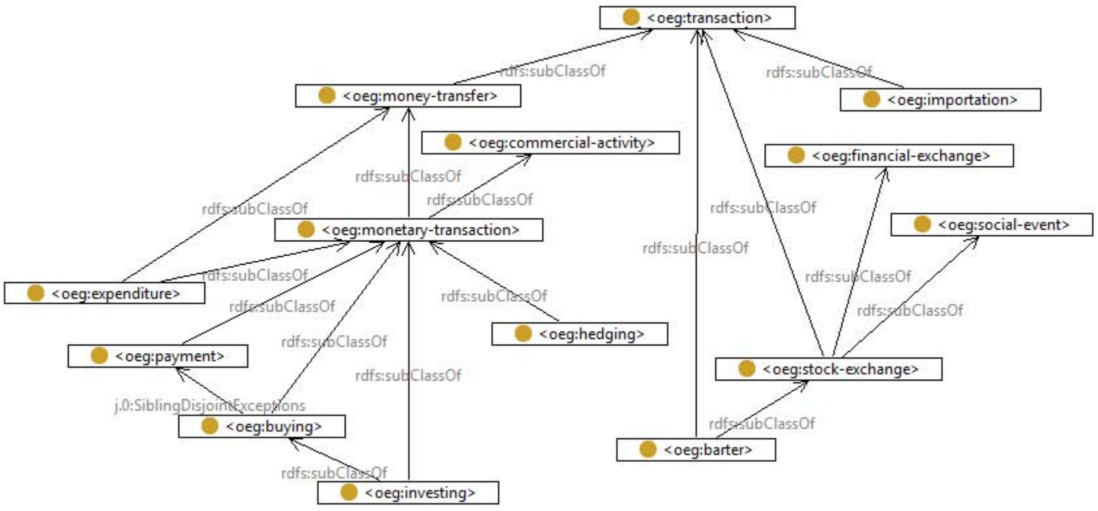


Figure 6.4: Excerpt from the financial ontology showing classes related to the Transaction class

Persons and Countries.

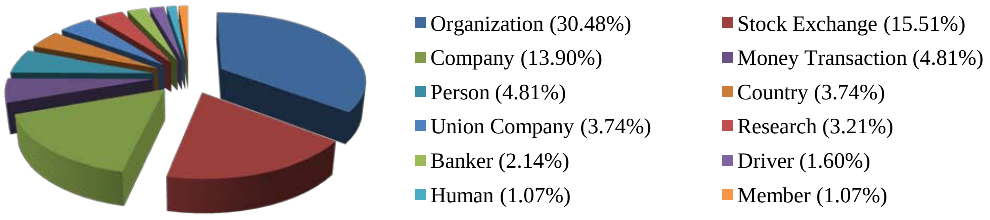


Figure 6.5: Percentage of classes in the ontology modules generated using PATO

Recall that we engaged four domain experts to evaluate the classes on each module. We have measured the reliability of the agreement between our evaluators using Fleiss' Kappa Fleiss (1971). This statistic measures how much the observed agreement exceeds the result that would be expected if all raters made their rates randomly. If a fixed number of people assign ratings to a number of items, then Kappa can be seen as a measure for the consistency of ratings. For the domain expert evaluations we obtained κ=0.137; a value greater than 0, meaning that the agreement exceeds the random result. Evaluators reached agreements by majority when rating 63.98% of the evaluated classes. Results of this evaluation are presented in table 6.9.

The overall precision of the ontology generated by our method was 80.67%[1]. The

[1]Evaluation data can be found in: http://delicias.dia.fi.upm.es/wiki/images/4/45/ FinanceOntoEval.zip

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. 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.

Module	**Classes**		**Module**	**Classes**	
	Number	**Precision**		**Number**	**Precision**
Organization	57	77,8%	Stock Exchange..	29	84.6%
Company	26	88.5%	Money Transaction	9	100%
Person	9	55.6%	Country	7	100%
Union (Company)	7	40%	Research	6	100%
Banker	4	100%	Driver	3	0%
Human	2	100%	Member	2	100%

Table 6.9: Precision of the ontology classes

With respect to the evaluation of the natural language sentences describing the relations in the ontology, the evaluators agreed in 70.33% of their ratings. Considering the sentences that evaluators were able to rate the 96.4% were valid relations. Finally, with respect to recall of the classes, 54% of the stock market classes in SUMO were covered by the output ontology. While the ontology include concepts for stocks and agents involved in the market transactions, it does not include information of other instruments such as bonds or annuaties. Nevertheless we cover in more detail some concepts such as money transactions. Please note that given our objective of developing a preliminary version of an ontology which allows practitioners to save time when developing an ontology, a recall of 54% means that we have covered almost half of the classes of a human created ontology in the domain.

6.3.4 Conclusions

We have presented an experiment where we evaluate, using precision and recall, the performance of the process to develop ontologies from user generated classification systems and linked data sets. We evaluated the intermediate results of the Terminology Extraction process as well as the final output produced by the Semantic Elicitation process. We found that Terminology Extraction process precision depends on the value of the threshold used to decide whether the spreading activation continuous processing more nodes in the graph. The higher the value of this threshold the more precise the terminology obtained. However this precise terminology implies a smaller number of terms. On the other hand, we evaluate the ontology produced in the Semantic Elicitation process first by analyzing the modules of classes contained in the ontology, and then by measuring precision and recall using a human-based evaluation. We found that the classes in the ontology were grouped in modules which most of the times were relevant to the stock market domain. In addition the human-driven evaluation of the classes and the relationships among them confirm that the ontology was produced with high precision: 80.67% precision for the classes, and 96.4% precision for the relations. In addition we compare the classes in the ontology to classes of a domain ontology and found that we cover 54% of the classes in the stock market domain.

We want to note that the evaluation of the generated ontology was challenging. Initially we considered an automatic evaluation using a gold standard domain ontology. Nevertheless to carry out this automatic evaluation we needed to match the labels of the ontology concepts with the labels of the golden standard ontology which do not necessarily have to be equal. Thus we were falling into an ontology matching problem (Euzenat and Shvaiko, 2007). Since we were not interested in solving this problem we decided to carry out the evaluation using domain experts. They had to asses the domain relevance of the classes and relationships making up the ontology. Given the fact that evaluators could have different opinions about the pertinence of a class to the domain, we required that each piece of information was assessed by different evaluators so that we can measure their agreement, and use only the decisions taken by majority.

In the context of this experiment we have achieved the thesis objectives **01**, **02**, and **03** through contributions **C1** and **C2**, and we have proven the hypothesis **H1**, **H2**, **H3**, and **H4**, which were described in chapter 3.

6.4 Survey: emergent semantics in Twitter lists

In this section we present a novel survey about the semantics of emergent relations in Twitter lists. These named lists represent a potentially rich source for harvesting knowledge, since they connect curators, members, subscribers and keywords appearing in list names. We explore which of such connections lead to emergent semantics and produce most related keywords. We obtain relations between keywords using the vector space model (Salton and Mcgill, 1986) and a topic modeling method, the Latent Dirichlet Allocation (Blei et al., 2003) LDA. Then we use metrics based on the WordNet synset structure (Fellbaum, 2005) to measure the semantic similarity between the related keywords obtained using the vector space model and LDA. In addition, we ground these keywords to linked data sets and present the relations found between them.

This survey is organized as follows. In section 6.4.1 we present the research framework, where we describe the models that we use for obtaining related keywords (see section 6.4.1.1), and the techniques that we used to characterise the semantics of the relations identified between keywords (see section 6.4.1.2). Next, in section 6.4.2, we present the experiment that we carried out to characterise the emergent semantics in Twitter lists along with the analysis of the results. Finally we present our conclusions in section 6.4.3.

6.4.1 Research framework

In this section we first present the models used for obtaining related keywords from Twitter lists. Then, we describe the techniques that we use to characterise the semantics of the obtained relations.

6.4.1.1 Related keywords in Twitter lists

We use the vector space model (Salton and Mcgill, 1986) to represent list keywords and their relationships with curators, members and subscribers. Each keyword is represented by three vectors of different dimension according to the type of relation represented. The use of vectors allows calculating similarity between them using standard measures such as the angle cosine.

Twitter lists can be defined as a tuple $TL = (C, M, S, L, K, R_l, R_k)$ where C, M, S, L, and K are sets of curators, members (of lists), subscribers, list names, and keywords respectively, $R_l \subseteq C \times L \times M$ defines the relation between curators, lists names, and members, and $R_k \subseteq L \times K$ represents keywords appearing in a list name. A list ϕ is defined as $(c, l, M_{c,l})$ where $M_{c,l} = \{m \in M | (c, l, m) \in R_l\}$. A subscription to a list can be represented then by $(s, c, l, M_{c,l})$. To represent keywords we use the following vectors:

- For the use of a keyword k according to curators we define $k_{curator}$ as a vector in $\Re^{\mathsf{C}}$ where entries in the vector $w_c = |\{(c, l, M_{c,l})|(l, k) \in R_k\}|$ correspond to the number of lists created by the curator c that contain the keyword k.

- For the use of a keyword k according to members we use a vector k_{member} in $\Re^{\mathsf{M}}$ where entries in the vector $w_m = |\{(c, l, m) \in R_l|(l, k) \in R_k\}|$ correspond to the number of lists containing the keyword k under which the member m has been listed.

- For the use of a keyword k according to subscribers we utilize a vector $k_{subscriber}$ in $\Re^{\mathsf{S}}$ where entries in the vector $w_s = |\{(s, c, l, M_{c,l})|(l, k) \in R_k\}|$ correspond to the number of times that s has subscribed to a list containing the keyword k.

In the vector space model we can measure the similarity between keywords calculating the cosine of the angle for the corresponding vectors in the same dimension. For two vectors k_i and k_j the similarity is $sim(k_i, k_j) = \frac{k_i \cdot k_j}{\|k_i\| \cdot \|k_j\|}$.

We also use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to obtain similar keywords. LDA is an unsupervised technique where documents are represented by a set of topics and each topic consists of a group of words. LDA topic model is an improvement over *bag of words* approaches including the vector space model, since LDA does not require documents to share words to be judged similar. As long as they share similar words (that appear together with same words in other documents) they will be judged similar. Thus documents are viewed as a mixture of probabilistic topics that are represented as a T dimensional random variable θ. For each document, the topic distribution θ has a Dirichlet prior $p(\theta|\alpha) \sim Dir(\alpha)$. In generative story, each document is generated by first picking a topic distribution θ from the Dirichlet prior and then use each document's topic distribution to sample latent topic variables z_i. LDA makes the assumption that each word is generated from one topic where z_i is a latent variable indicating the hidden topic assignment for word w_i. The probability of choosing a word w_i under topic z_i, $p(w_i|z_i, \beta)$, depends on different documents.

We use the bag of words model to represent documents as input for LDA. For our study keywords are documents and words are the different users according to their role in the list structure. To represent keywords we use the following sets:

- For a keyword k according to curators we use the set $k_{bagCurator} = \{c \in C|(c, l, m) \in R_l \wedge (l, k) \in R_k\}$ representing the curators that have created a list containing the keyword k.

- For a keyword k according to members we use a set $k_{bagMember} = \{m \in M|(c, l, m) \in R_l \wedge (l, k) \in R_k\}$ corresponding to the users who have been classified under lists containing the keyword k.

- For a keyword k according to subscribers we use a set $k_{bagSubscriber} = \{s \in S|(s, c, l, M_{c,l}) \wedge (l, k) \in R_k\}$, that is the set of users that follow a list containing the keyword k.

LDA is then executed for all the keywords in the same representation schema (*i.e.*, based on curators, members, or subscribers) generating a topic distribution θ for each document. We can compute similarity between two keywords k_i and k_j in the same representation schema by measuring the angle cosine of their corresponding topic distributions θ_i and θ_j.

6.4.1.2 Characterising relations between keywords

To investigate the relevance and type of the relations between keywords obtained from twitter lists we use state of the art similarity measures based on WordNet. In addition, given the limited scope of WordNet we complement our study using SPARQL queries on knowledge bases published as linked data.

WordNet-based Similarity To validate the relations found from keyword co-occurrence analysis in Twitter lists, we use similarity measures that tap into WordNet (Fellbaum, 2005). WordNet is a lexical database where synonyms are grouped on synsets, with each synset expressing a concept.

A natural measure of similarity between words is the length of the path connecting the corresponding synsets (Jiang and Conrath, 1997; Pedersen et al., 2004). The shorter the path the higher the similarity. This length is usually calculated in the noun and verb is-a hierarchy according to the number of synsets in the path connecting the two words. In the case of two synonyms, both words belong to the same synset and thus the path length is 1. A path length of 2 indicates an is-a relation. For a path length of 3 there are two possibilities: (*i*) both words are under the same hypernym known as *common subsumer*, and therefore the words are siblings, and (*ii*) both words are connected through an in-between synset defining an indirect is-a relation. Starting with 4 the interpretation of the path length is harder.

However, the weakness of using path length as a similarity measure in WordNet is that it does not take into account the level of specificity of synsets in the hierarchy. For instance, *measure* and *communication* have a path length of 3 and share *abstraction* as a common subsumer. Despite low path length, this relation may not correspond to the human concept of similarity due to the high level of abstraction of the concepts involved.

Abstract synsets appear in the top of the hierarchy, while more specific ones are placed at the bottom. Thus, Wu and Palmer (1994) propose a similarity measure which includes the depth of the synsets and of the least common subsumer (see equation 6.1). The least common subsumer *lcs* is the deepest hypernym that subsumes both synsets, and depth is the length of the path from the root to the synset. This similarity range between 0 and 1, the larger the value the greater the similarity between the terms. For terms *measure* and *communication*, both synsets have depth 4, and the depth of the *lcs* *abstraction* is 3; therefore, their similarity is 0.75.

$$wp(synset_1, synset_2) = 2 * depth(lcs)/(depth(synset_1) + depth(synset_2) \tag{6.1}$$

Jiang and Conrath (1997) propose a distance measure that combines hierarchical and distributional information. Their formula includes features such as local network density (*i.e.*, children per synset), synset depth, weight according to the link type, and information content IC of synsets and of the least common subsumer. The information content of a synset is calculated as the inverse log of its probability of occurrence in the WordNet hierarchy. This probability is based on the frequency of words subsumed by the synset. As the probability of a synset increases, its information content decreases. Jiang and Conrath distance can be computed using equation 6.2 when only the information content is used. A shorter distance means a stronger semantic relation. The IC of *measure* and *communication* is 2.95 and 3.07 respectively while *abstraction* has a IC of 0.78, thus their semantic distance is 4.46.

$$jc(synset_1, synset_2) = IC(synset_1) + IC(synset_2) - 2 * IC(lcs) \tag{6.2}$$

Linked Data to Identify Relation Types WordNet-based analysis is rather limited, since WordNet contains a small number of relations between synsets. To overcome this limitation and improve the detection of relationships, we use general purpose knowledge bases such as DBpedia (Bizer et al., 2009b), OpenCyc,[1] and UMBEL[2], which provide a wealth of well-defined relations between concepts and instances.

Our aim is to bind keywords extracted from list names to semantic resources in these knowledge bases so that we can identify which kind of relations appear between them. To do so we harness the high degree of interconnection in the linked data cloud offered by DBpedia. We first ground keywords to DBpedia (García-Silva et al., 2009), and then we browse the linked data set for relations connecting the keywords.

After connecting keywords to DBpedia resources we query the linked data set to search for relations between pairs of resources. We use a similar approach to (Heim et al., 2010) where SPARQL queries are used to search for relations linking two resources r_s and r_t. We define the path length L as the number of objects found in the path linking r_s with r_t. For $L = 2$ we look for a $relation_i$ linking r_s with r_t. As we do not know the direction of $relation_i$, we search in both directions: 1) r_s $relation_i$ r_t, and 2) r_t $relation_i$ r_s. For $L = 3$ we look for a path containing two relationships and an

[1]OpenCyc home page: http://sw.opencyc.org/
[2]UMBEL home page: http://www.umbel.org/

intermediate resource *node* such as: r_s $relation_i$ $node$, and $node$ $relation_j$ r_t. Note that each relationship may have two directions and hence the number of possible paths is $2^2 = 4$. For $L = 4$ we have three relationship placeholders and the number of possible paths is $2^3 = 8$. In general, for a path length L we have $n = \sum_{l=2}^{L} 2^{(l-1)}$ possible paths that can be traversed by issuing the same number of SPARQL queries[1] on the linked data set.

For instance, let us find the relation between the keywords *Anthropology* and *Sociology*. First both keywords are grounded to the respective DBpedia resources, in this case *dbpr:Anthropology* and *dbpr:Sociology*. Figure 6.6 shows linked data relating these DBpedia resources. To retrieve this information, we pose the query shown in Listing 6.1.[2] The result is the triples making up the path between the resources. In our case we discard the initial *owl:sameAs* relation between DBpedia and OpenCyc resources, and keep the assertion that Anthropology and Sociology are Social Sciences.

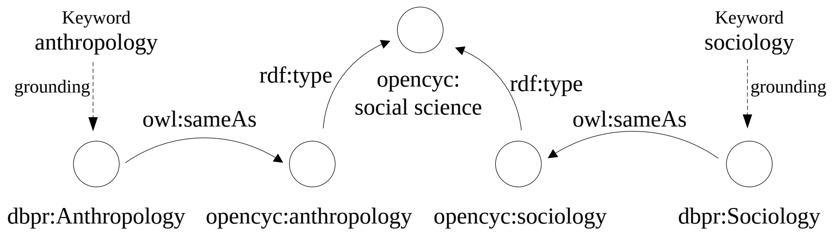


Figure 6.6: Linked data showing the relation between the anthropology and sociology

```
SELECT *
WHERE{<dbpr:Anthropology> ?relation1 ?node1. ?node1 ?relation2 ?node2.
      <dbpr:Sociology> ?relation4 ?node3. ?node3 ?relation3 ?node2. }
```

Listing 6.1: SPARQL query for finding relations

6.4.2 Experiment

To analyse the emergent semantics from Twitter lists we carried out an experiment where we obtained related keywords from a data set extracted from Twitter and characterised the relations between those keywords using similarity measures based on WordNet and SPARQL queries over a data set of linked data.

[1]Note that for large L values the queries can last long time in large data sets.
[2]Property paths, in SPARQL 1.1 specification, allow simplifying these queries.

6.4.2.1 Data set

Twitter offers an Application Programming Interface (API) for data collection. We collected a snowball sample of users and lists as follows. Starting with two initial seed users, we collected all the lists they subscribed to or are members of. There were 260 such lists. Next, we expanded the user layer based on current lists by collecting all other users who are members of or subscribers to these lists. This yielded an additional set of 2573 users. In the next iteration, we expanded the list layers by collecting all lists that these users subscribe to or are members of. In the last step, we collected 297,521 lists under which 2,171,140 users were classified. The lists were created by 215,599 distinct curators, and 616,662 users subscribe to them[1]. From list names we extracted, by approximate matching of the names with dictionary entries, 5932 unique keywords; 55% of them were found in WordNet. The dictionary was created from article titles and redirection pages in Wikipedia.

6.4.2.2 Set up

For each keyword we created the vectors and the bags of words for each of the three user-based representations defined in section 6.4.1.1. We calculated cosine similarity in the corresponding user-based vector space. We also run the LDA algorithm over the bags of words and calculated the cosine similarity between the topic distribution produced for each document. We kept the 5 most similar terms for each keyword according to the Vector-space and LDA-based similarities.

6.4.2.3 WordNet analysis

For each pair of similar keywords we calculated their similarity according to Jiang and Conrath (JC) and Wu and Palmer (WP) formulas. To gain an initial insight about these measures we calculate the correlation between them (see Figure 6.7). We use the Pearson's coefficient of correlations which divides the covariance of the two variables by the product of their standard deviations.

In general these results show that Vector-space and LDA similarity based on members produce the most similar results to that of WordNet measures. Vector-space similarity based on subscribers and curators also produces correlated results, although significantly lower. LDA similarity based on subscribers results is correlated to JC distance but not to WP similarity. Finally LDA based on curators produces results that are not correlated to WordNet similarities.

[1]The data set can be found here: `http://goo.gl/vCYyD`

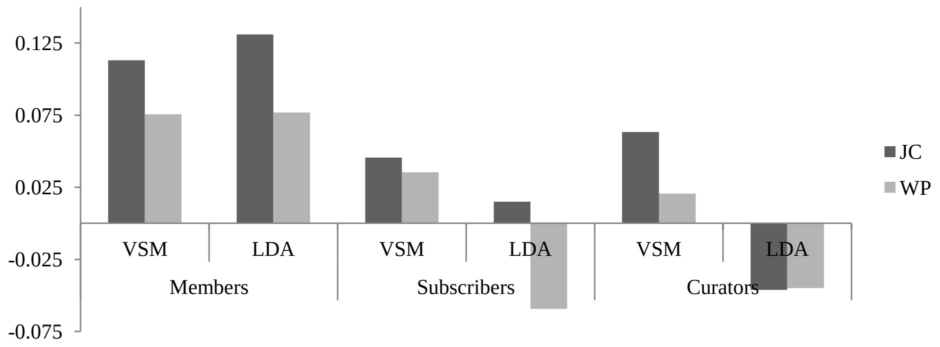


Figure 6.7: Coefficient of correlation between Vector-space and LDA similarity with respect to WordNet measures

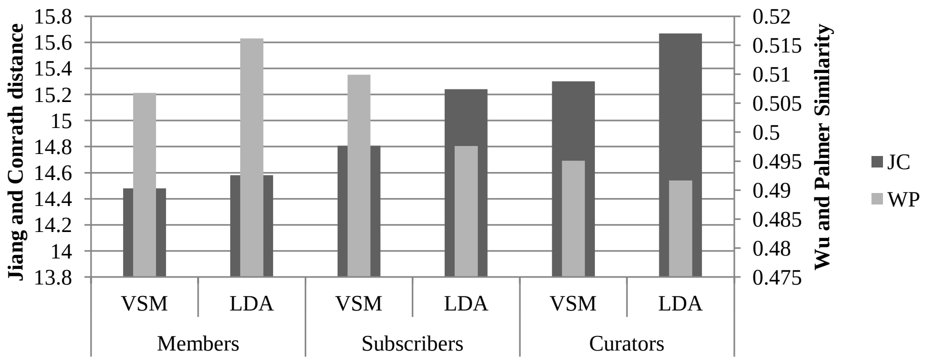


Figure 6.8: Average Jiang and Conrath distance and Wu and Palmer similarity

Correlation results can be partially explained by measuring the average of JC distance and WP similarity[1] (see figure 6.8). Vector-space and LDA similarities based on Members have the shortest JC distance, and two of the top tree WP similarity values. Vector-space similarity based on subscribers has also a short JC distance, and a high WP similarity. For the rest of similarities JC distances are longer and WP similarity lower.

To identify the type of relations found by Vector-space and LDA similarities we calculate, as shown in table 6.10, the path length of the corresponding relations in WordNet. To guarantee a base similarity, we use a threshold of 0.1; similarities under this value were discarded. Note that in WordNet different part of speech categories have distinct hierarchies and hence the path length can be calculated only for terms in the same category. According to the path length, the similarity based on members produce the highest number of synonyms (path length=1), reaching a 10.87% of the

[1]The averages were calculated over relations for which both terms were in WordNet.

relations found in WordNet for the case of LDA similarity. In this case, the LDA model analyzes co-occurrence of groups of members across different keywords to identify related keywords. Unlike the vector space model, which requires exact members to be present in similar keywords, LDA allows synonyms, i.e., different members that tend to co-occur with the same sets of keywords, to contribute to keyword similarity.

Table 6.10: Path length in WordNet for similar Keywords according to Vector-space and LDA models.

Path Length	**Members**		**Subscribers**		**Curators**	
	VSM	**LDA**	**VSM**	**LDA**	**VSM**	**LDA**
1	**8.58%**	**10.87%**	3.97%	3.24%	1.24%	0.50%
2	**3.42%**	**3.08%**	1.93%	0.47%	0.70%	0.00%
3	2.37%	**3.77%**	2.96%	2.06%	2.38%	**4.03%**
>3	67.61%	65.50%	67.27%	67.56%	77.83%	75.81%

Similarity based on subscribers and curators produce a significative lower number of synonyms. Likewise, similarity based on members produces the highest number of direct is-a relations (path length=2). LDA similarity based on curators produce the highest number of keywords directly related by a common superclass or an indirect is-a relation (path length=3).

Given that the majority of relations found in WordNet have a path length greater than or equal to 3, we decided to categorize them according to whether the relation is based on a common subsumer or whether it is based on linked is-a relations. In average 97.65% of the relations with a path length ≥ 3 involve a common subsumer.

As it was argued before, the depth of the least common subsumer influences the relevance of a relation. A manual inspection of the WordNet hierarchy shows that synsets being at a distance greater than or equal to 5 from the root may be considered as more specific. Figure 6.9 shows the percentage of relations according to the depth of the least common subsumer in the WordNet hierarchy. For a depth of the LCS greater than or equal to 5 and to 6 the Vector-space similarity based on subscribers produces the highest percentage of relations (39.19% and 20.62% for each case) followed by the Vector-space similarity based on members (37.07% and 17.96%). Starting from a depth of the LCS greater than or equal to 7 until 9 the LDA and Vector-space similarity based on members gathers the highest percentage of relations.

In addition to the depth of the LCS, the other variable to explore is the length of the path setting up the relation. The stacked columns in figure 6.10 show the cumulative

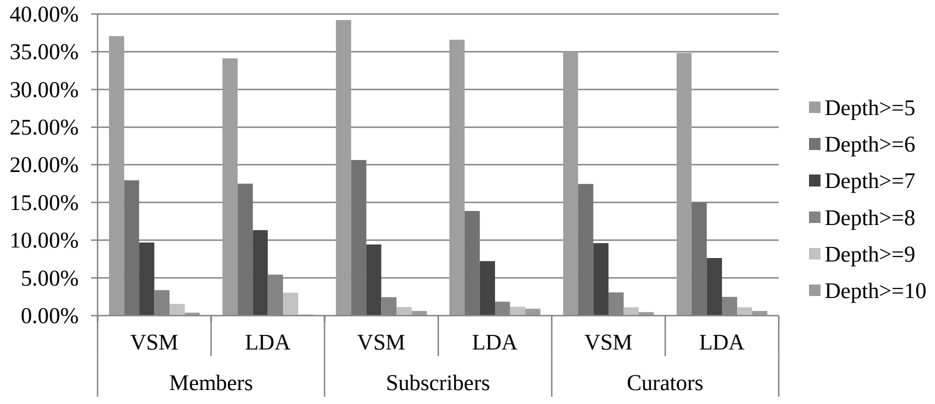


Figure 6.9: Relations according to the depth of the least common subsumer LCS.

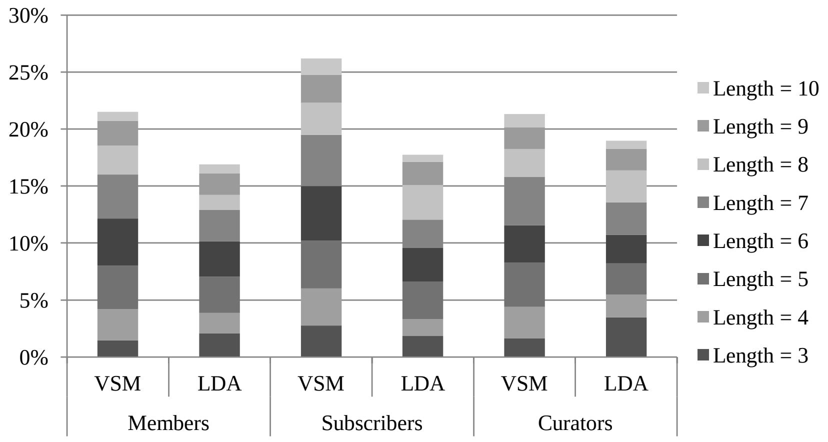


Figure 6.10: Relations according to the path length for those cases where the least common subsumer has depth greater or equal to 5.

percentage of relations found by Vector-space and LDA models according to the path length of the relation in WordNet, with a depth of the least common subsumer greater than or equal to 5. From the chart we can state that Vector-space similarity based on subscribers produces the highest percentage of relations (26.19%) with a path length ≤ 10. This measure also produces the highest percentage of relations for path lengths ranging from 9 to 4. The Vector-space similarity based on members produces the second highest percentage of relations for path lengths from 10 to 6.

In summary, we have shown that similarity models based on members produce the results that are most directly related to the results of similarity measures based on WordNet. These models find more synonyms and direct relations is-a when compared to the models based on subscribers and curators. These results suggest that some users are classified under different lists named with synonyms or with keywords representing a concept in a distinct level of specificity. We also discovered that the majority of relations found by any model have a path length ≥ 3 and involve a common subsumer. Vector-space model based on subscribers produces the highest number of relations that can be considered specific (depth of LCS ≥ 5 or 6). However, for more specific relations ($7 \leq$ depth of LCS ≤ 9) similarity models based on members produce a higher number. In addition we considered the path length, for those relations containing a LCS placed in a depth ≥ 5 in the hierarchy, as a variable influencing the relevance of a relation. Vector-space model based on subscriber finds the highest number of relations with $4 \leq$ length ≤ 10. In general similarity models based on curators produce a lower number of relations. We think this may be due to the scarcity of lists per curator. In our dataset each curator has created 1.38 lists in average.

6.4.2.4 Linked Data analysis

Our approach found DBpedia resources for 63.77% of the keywords extracted from Twitter Lists. In average for the 41.74% of relations we found the related keywords in DBpedia. For each relation found by Vector-space or LDA similarity we query the linked data set looking for patterns between the related keywords. Figure 6.11 shows the results according to the path length of the relations found in the linked data set. These results are similar to the ones produced by WordNet similarity measures. That is, similarity based on Members produce the highest number of synonyms and direct relations though in this case Vector-space similarity produces more synonyms than LDA. Vector-space similarity based on subscribers has the highest number of relations of length 3, followed by Vector-space and LDA similarity based on members.

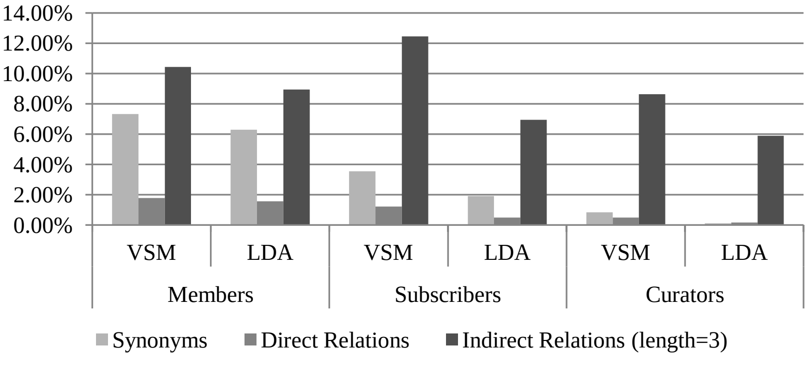


Figure 6.11: Relations identified from linked data queries.

Given that the Vector-space model based on members found the majority of direct relations, we present, in table 6.11, the relations identified in the linked data set. *Broad term* and *subClassOf* are among the most frequent relations. This means that members of lists are usually classified in lists named with keywords representing a concept with a different level of specificity. Other relations that are difficult to elicit from traditional lexicons are also obtained, such as *developer*, *genre* or *largest city*.

In addition we also investigate the type of relations of length 3 elicited using the Vector-space model based on subscribers. The most common patterns found in the linked data set were $r_s \overset{relation_1}{\rightarrow} object \overset{relation_2}{\leftarrow} r_t$, and $r_s \overset{relation_1}{\leftarrow} object \overset{relation_2}{\rightarrow} r_t$ with 54.73% and 43.49% of the relations respectively. Table 6.12 shows the obtained relations according to each pattern.

With respect to the first pattern, 97.96% of the related keywords can be considered siblings since they are associated via *typeOf* or *subClassOf* relations with a common class. That is, some subscribers follow lists that share a common super concept. On the other hand, the second pattern shows a wider range of relations. Keywords are related since they are *genres*, *occupations*, *products*, *industries*, or *main interest* that appear together in the description of an individual in the linked data set.

Table 6.11: Direct relations established by the vector-space model based on members.

Relation type		**Example of keywords**	
Broader Term	26%	life-science	biotech
subClassOf	26%	authors	writers
developer	11%	google	google_apps
genre	11%	funland	comedy
largest city	6%	houston	texas

6.4.3 Conclusions

We have evaluated different models to elicit semantic relations from Twitter lists. These models represent keyword co-occurrence in lists based on three user roles: curators, subscribers and members. We measure similarity between keywords using the vector-space model and a topic based model known as Latent Dirichlet Allocation. Then we use Wordnet similarity measures including Wu and Palmer, and Jiang and Conrath distance, to compare the results of the vector-space and LDA models.

Results show that applying vector-space and LDA metrics based on members produce the most correlated results to those of WordNet-based metrics. We found that

Table 6.12: Indirect relations of length 3 found in the linked data set for the relations established by the vector-space model based on subscribers

$r_s \overset{relation_1}{\rightarrow} object \overset{relation_2}{\leftarrow} r_t$			
Relations			**Example**
type	type	67.35%	nokia → company ← intel
subClassOf	subClassOf	30.61%	philanthropy → activities ← fundraising

$r_s \overset{relation_1}{\leftarrow} object \overset{relation_2}{\rightarrow} r_t$			
Relations			**Example**
genre	genre	12.43%	theater ← Aesthetica → film
genre	occupation	10.27%	fiction ← Adam Maxwell → writer
occupation	occupation	8.11%	poet ← Alina Tugend → writer
product	product	7.57%	clothes ← ChenOne → fashion
product	industry	9.73%	blogs ← UserLand Software → internet
occupation	known for	5.41%	author ← Adeline Yen Mah → writing
known for	known for	3.78%	skeptics ← Rebecca Watson → atheist
main interest	main interest	3.24%	politics ← Aristotle → government

these measures produce relations with the shortest Jiang and Conrath distance and high Wu and Palmer similarities. In addition, we categorize the relations found by each model according to the path length in WordNet. Models based on members produce the highest number of synonyms and of direct is-a relations. However, most of the relations have a path length ≥ 3 and have a common subsumer. We analyze these relations using the depth of the LCS and the path length as variables that help to identify the relevance of relations. This analysis shows that the vector-space model based on subscribers finds the highest number of relations when relevance is defined by a depth of LCS ≥ 5, and the path length of relations is between 10 and 4.

We also investigate the type of relations found by each of the models using general knowledge bases published as linked data. We categorize the relations elicited by each model according to the path length in the linked data set. These results confirm that the models based on members produce the highest number of synonyms and direct relations. In addition, we find that direct relations obtained from models based on members are mostly *Broader Term* and *subclassOf*. Finally, we study the type of relations obtained from the vector-space model based on subscribers with a path length of 3 and find that mostly they represent sibling keywords sharing a common class, and subjects that are related through an individual.

In the context of this experiment we have achieved the thesis objective **04** through contribution **C1**, and we have proven the hypothesis **H5**, **H6** and **H7**, which were described in chapter 3.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The active involvement of users in the generation of content on the Web 2.0 has led to the creation of a massive amount of information resources that need to be organized so that they can be better retrieved and managed. Different strategies have been used to overcome this information overload problem, including the use of tags to annotate resources in *folksonomies*, and the use of *lists* or collections to organize them. The bottom-up nature of these *user-generated classification systems*, as opposed to systems maintained by a small group of experts, have made them interesting sources for acquiring knowledge.

In parallel with the Web 2.0 a new *Web of data* has been developed as part of the Semantic Web (Berners-Lee et al., 2001), which has emerged from the creation of data sets published following the linked data principles (Bizer et al., 2009a). Some of these datasets are large general purpose ontologies such as DBpedia, OpenCyc, WordNet, UMBEL, and YAGO which have been created manually or automatically by extracting knowledge from unstructured and structured resources such as Wikipedia and WordNet. The fact that these ontologies are interconnected among them and can be accessed through a unique query language turn them into interesting resources to reuse existing conceptualisations.

Thus, we framed this thesis work in a global objective of *researching novel methods and techniques to automatically develop domain ontologies leveraging user-generated classification systems and the Web of data.* Ontologies are a cornerstone of the Semantic Web since they can provide semantics to raw data so that these data can be consumed automatically by software agents. However building ontologies is a complex process where ontology engineers face the so-called knowledge acquisition bottleneck. That is, they have to obtain knowledge from domain experts or domain relevant resources including standards and existing taxonomies to represent it in the form of an ontology. Thus, we aim at automatically developing an initial version of an ontology from user-generated classification systems and existing ontologies in the Web of Data so that ontology engineers do not have to start from scratch when developing domain ontologies.

When surveying the state of the art we identified two open research problems. First, we found that different methods have been proposed to extract ontologies from folksonomies. However from our survey we concluded that current methods produce shallow and limited ontologies where, in some cases, there was no distinction between classes and instances, neither there was a clear definition of the relations in the ontology. In addition we found that none of the surveyed approaches produce domain ontologies. That is in these approaches they do not narrow the scope of the produced ontology to relevant concepts in a domain. Hence we stated that *the development of a method, and its supporting techniques, to automatic elicit formal domain ontologies from folksonomies is still an open research problem* (see section 2.3.9). Second, we found that user-generated lists were not used in knowledge acquisition processes and there was not any quantitative or qualitative survey about the emergent semantics which can be extracted from them. Therefore we considered that *there is a lack of a survey about the semantics that can be elicited from user-generated lists.* Such surveys lay the foundations over which knowledge acquisition processes can be built since they identify the amount and the type of information that can be obtained from knowledge sources, in this case user-generated lists.

To cope with these open research problems we defined objectives, and hypotheses on which the contributions to achieve these objectives rely. We also carried out some experiments where we verify the hypotheses through the evaluation of the method developed and techniques adapted in this thesis. In the following we describe in short the objectives, hypotheses, contributions and the experiments associated with each of the two open research problems.

Ontology development from user-generated classification systems

With respect to the development of a method to automatic elicit formal domain ontologies from folksonomies relying on linked data sets we define the following objectives:

O1. To propose methods and techniques to leverage the knowledge in user-generated classification systems in the ontology development.

O2. To propose methods and techniques to reuse ontologies published as linked data in the ontology development process.

O3. To propose an integrated method to create ontologies relying on folksonomies and linked data.

We rely on the following hypotheses to achieve the aforementioned objectives.

H1. User-generated classification systems can be mined to collect knowledge pertinent to a domain so that this knowledge can be used in an ontology development process.

H2. Ontologies published as linked data can be used to make explicit and formal the knowledge extracted from user-created classification systems.

H3. It is possible to create a method, relying on the hypotheses H3 and H4, to create a domain ontology.

H4. It is possible to use existing techniques to automatically carry out the processes proposed in the method.

Based on these hypotheses and with the aim to achieve the objectives that we have posed in this thesis we have generated the following contributions:

C1. **An integrated method to develop an ontology schema from user-generated classification systems relying on linked data**. This method covers different stages of the ontology development process:

- Elicitation of domain terminology by collecting relevant terms from user-generated classification systems.
- Identification of classes from the extracted terminology by reusing existing classes of ontologies published following the linked data principles.
- Discovery of relations between the identified classes by reusing existing relations of ontologies published following the linked data principles.

C2. **A set of techniques to support the distinct activities and tasks proposed in the method**. We have adapted the following techniques to achieve the goals of each of the processes.

- Spreading activation for eliciting domain terminology.

- Vector space model and dynamic queries in SPARQL for identifying classes.
- Dynamic queries in SPARQL for discovering relations.

We designed different experiments to evaluate the method and techniques proposed to generate domain ontologies and to verify the hypotheses we posed in this thesis. Thus in section 6.1 we evaluated the Normalization task which is part of the Data Preprocessing Activity of the Terminology Extraction process. Similarly, in section 6.2 we evaluated the Semantic Grounding activity which is part of the Semantic Elicitation process. Next, we evaluated in section 6.3 the whole approach, which comprises contributions C1 and C2, to obtain an ontology in the stock market domain from a folksonomy generated in the Delicious bookmarking site and reusing the DBpedia, OpenCyc and UMBEL ontologies. The results of this experiment where satisfactory in terms of precision and recall of the generated ontology and therefore in this context we verified hypothesis H1, H2, H3 and H4.

Emergent semantics from user-generated lists

Regarding the survey of the semantics that can be elicited from user generated lists we define the following objective:

O4. To analyse the emerging semantics of Twitter lists, a user-generated classification system which has not been studied as source of knowledge yet.

To achieve this objective we rely on the following hypotheses:

H5. There is an emergent semantics which result of the aggregation of the individual classifications in user-generated lists.

H6. It is possible to use existing similarity measures based on WordNet to analyze the emergent semantics in user-generated lists.

H7. It is possible to use existing ontologies published as linked data to analyze the emergent semantics in user-generated lists.

Based on these hypotheses and with the aim to fulfill the objective we have generated the following contribution:

C3. **A survey of the emergent semantics in Twitter lists**. We presented a survey of the emerging semantics in Twitter lists where we quantify and characterise the knowledge that can be extracted from this user-generated lists. To do so we rely on the vector space model and the Latent Dirichlet Allocation to elicit related terms from this user-generated list. Then we use similarity measures based on WordNet and queries over linked data sets to characterise the semantic of the relations between the related terms.

To survey this user-generated list we carried out an experiment, presented in section 6.4, where we were able to verify the hypotheses H5, H6, and H7.

Future Work

While developing this thesis we were identifying new opportunities and research problems that did not fall into the scope that we have defined for this thesis work and therefore we decided to postpone them. In the following we present our future work:

- **Apply the method generated in this thesis to user-generated lists.** In this thesis we have shown that it is possible to elicit keywords from user-generated lists which are related semantically. Since user-generated lists are user-generated classification systems we can apply the method to obtain ontologies from them relying on the Web of Data.

- **Survey the emergent semantics of user-generated lists produced in novel applications.** In addition to Twitter, other web applications have started allowing users to create lists. For instance in Pinterest[1] and Flickr[2] users create lists to arrange pictures, while in Delicious[3] they do it to organize bookmarks. These lists can be shared and used by other users in the platform and therefore we can benefit not only of the individual contribution of the list creator but of the aggregated use given by the community. Thus, these user-generated classification systems are novel sources for which an emergent semantics can be characterised. According to the results obtained in this charaterisation we could apply the method to obtain ontologies developed in this thesis.

- **Multilingual ontologies**. The user-generated classification systems in which we are interested are created by large user communities in Web applications with a worldwide scope, therefore reaching users with different languages. These classification systems contain names (*i.e.*, tags and list names) defined by users which are

[1]see `http://pinterest.com`
[2]http://www.flickr.com
[3]http://delicious.com/

written in their own languages. This fact turns user-generated classification systems into knowledge sources from which multilingual information can be elicited. Thus a future line of research is to exploit these classification systems to obtain multilingual ontologies.

- **Using ontologies to improve recommendation and search functionalities in web applications relying on user-generated classification systems.** We can improve recommendation and search functionalities in web applications from which we are obtaining the ontologies. In folksonomies, tag recommendation can be based on the ontological relations associated with the concept representing the tag meaning. For instance, when the user type a tag the system can suggest superclasses and object properties related to the concept representing that tag in the ontology. In case of user-generated lists a resource recommendation system can suggest for a given list other resources which are classified under lists with names which are synonyms or superclasses of that list. On the other hand, search processes may benefit of the ontologies so that they can carry out i) query expansion processes based on superclasses and object properties of the class representing the keyword used in the query, and ii) query disambiguation by asking or identifying automatically the meaning (*i.e.*, the ontology concept) of the keyword used in the query and use the classes in the ontological context to pose less ambiguous queries.

ANNEX A

ONTOLOGY FOR DESCRIBING THE METHOD DATA

In this annex we present the ontology (see figure A.1) designed to model the input and output data of the processes, activities and tasks making up the method. This ontology[1], developed in OWL-DL [2], was designed by reusing existing ontologies, and following ontology design best practices defined by the World Wide Web Consortium W3C such as the part-whole best practices which are described later in this section.

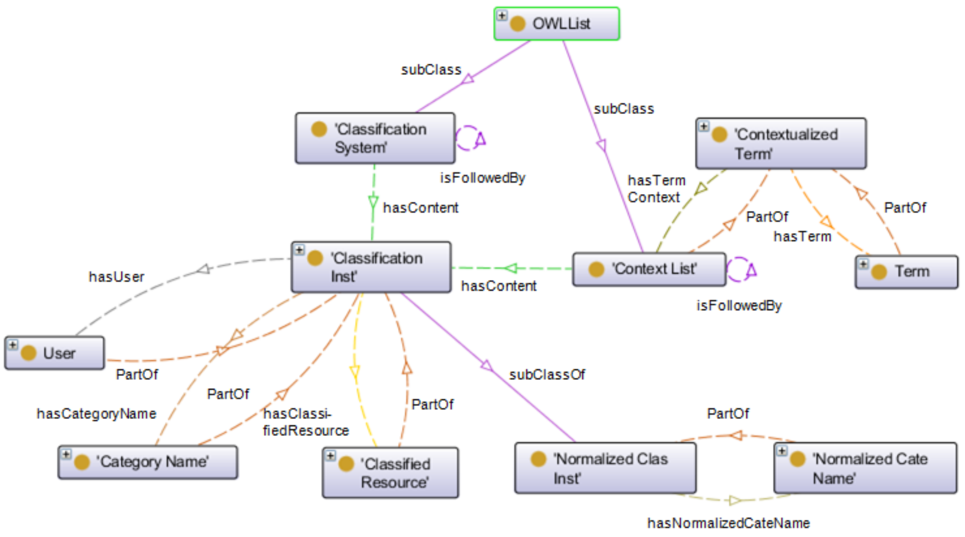


Figure A.1: Ontology modeling the data structures presented in the method.

[1] The ontology is available at `http://www.oeg-upm.net/index.php/ontologies/288-ontologies-user-generated-classification-systems`

[2] OWL specification is available at: `http://www.w3.org/TR/owl-guide/`

The classification system is represented by the class *ClassificationSystem* which is modeled as a list of classification instances, which in turn are represented by the class *ClassificationInst.* To model a list we reuse the ontology OWLlist [1] which has been proposed in Drummond et al. (2006). In this ontology a list is a sequence of nodes where each node contains data and a link to the next node in the list. Thus to model the list a node class is created as a subclass of OWLlist, and then the *hasContent* and *isFollowedBy* object properties have to be defined. While the hasContent relates the node with the class containing the data, the isFollowedBy defines the class of the next node in the list. Therefore in the case of the ClassificationSystem class we defined it as *subClassOf* OWLlist and set the hasContent property to a ClassificationInstance class and the isFollowedBy property to a ClassificationSystem class.

A classification instance consists of a user, a category name and classified resource which are represented in the ontology by the corresponding classes. The *consists of* relation has been modeled in two ways. first we have created the object properties *hasUser*, *hasCategoryName* and *hasClassifiedResource* to specify the classification instance parts. On the other hand, we have modeled the *partOf* relation between the constituent parts and the classification instance following the best practices in this respect provided by the W3C [2]. One of the best practices recommends to define the partOf as a taxonomy. To do so, first the partOf is defined as an object property. Then each of the constituent parts has to be declared as subclass of a restriction over the range of values that the PartOf can take, which in this case is ClassificationInst. As we require that all users, category names and classified resources are part of at least one classification instance, we create the restriction using an existential quantifier. In addition we require that users, category names, and classified resources only can be part of classification instances, and hence we create another restriction using an universal quantifier. All the partOf relations presented in this ontology follows the recommend best practice and use both, existential and universal quantifiers when defining the restriction over the range of the partOf object property.

A normalized classification instance, represented by the class *NormalizedClasInst*, is a classification instance plus a normalized category name, which is represented by the class *NormalizedCateName.* Therefore the NormalizedClasInst class is subclass of ClassificationInst, and is related to the class NormalizedCateName by means of the object property *hasNormalizedCateName.* In addition, the normalizedCateName class

[1]OWLlist is available in `http://www.co-ode.org/ontologies/lists/2008/09/11/list.owl`

[2]Part-whole best practices published in : `http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/simple-part-whole-relations-v1.3.html`

is defined as *partOf* of the NormalizedClasInst class.

A contextualized term, represented by the class *ContextualizedTerm*, consists of a term and its context, which in turn are represented by the class *Term* and *ContextList*. The *consists of* relation has been modeled using the object properties *hasTerm* and *hasTermContext*. On the other hand, we have defined a *partOf* relation between the constituent parts Term and ContextList, and the ContextualizedTerm class. The ContextList is a list of classification instances. Therefore it is defined a subclass of OWLlist. its hasContent and isFollowedBy object properties are set to the class ClassificationInstance and to the class ContextList respectively. Finally, we have included an standalone *TransformationInstance* class to represent the instances which has been transformed in the Data Preprocessing activity of the method. Note that is not possible to define in advance the semantics of this class since it depends on the data structure which is defined during the execution of the method. And therefore the ontology engineer must adapt this ontology according to the circumstances defined by the method execution.

In the following we present the definitions of the the different classes and object properties making up the ontology.

A.1 Classes

OWLList

This class is imported from the ontology list.owl and it represents a lists. The list is defined by specifying the content of each node in the list (hasContent object property) and a link to the next node in the list (isFollowedBy object property).

ClassificationSystem

This class represents the classification systems as a list of classification instances. It is defined as:

ClassificationSystem $\sqsubseteq$ OWLList
ClassificationSystem $\sqsubseteq \forall$ hasContents ClassificationInst
ClassificationSystem $\sqsubseteq \forall$ isFollowedBy ClassificationSystem

ClassificationInst

This class represents a classification instance which is an individual classification of a resource in the classification system. That is, the relation between a user who has used a category name to classify a resource.

User

This class represents a user in the classification system. A user is, at least and only, part of a classification instance. In addition users, category names, and classified resources are disjoint. It is defined as:

User $\sqsubseteq \forall$ partOf ClassificationInst
User $\sqsubseteq \exists$ partOf ClassificationInst
User $\sqsubseteq \neg$ CategoryName
User $\sqsubseteq \neg$ ClassifiedResource

ClassifiedResource

This class represents a classified resource (e.g., Web pages, pictures, and users) that users classify in the system. These resources are, at least and only, part of a classification instance. In addition, classified resources, category names and users are disjoint. It is defined as:

ClassifiedResource $\sqsubseteq \forall$ partOf ClassificationInst
ClassifiedResource $\sqsubseteq \exists$ partOf ClassificationInst
ClassifiedResource $\sqsubseteq \neg$ CategoryName
ClassifiedResource $\sqsubseteq \neg$ User

CategoryName

This class represents the category names (e.g., tags or list names) that users assign to classify resources in the system. These category names are, at least and only, part of a classification instance. In addition, category names, classified resources, and users are disjoint. It is defined as:

CategoryName $\sqsubseteq \forall$ partOf ClassificationInst
CategoryName $\sqsubseteq \exists$ partOf ClassificationInst
CategoryName $\sqsubseteq \neg$ ClassifiedResource
CategoryName $\sqsubseteq \neg$ User

NormalizedClasInst

This class represents the normalized version of a classification instance, where the category name has been turn into a normalized category name. This class is subclass of the ClassificationInst class. It is defined as:

NormalizedClasInst $\sqsubseteq$ ClassificationInst

NormalizedCateName

This class represents the normalized version of a category name. A normalized category name is, at least and only, part of a normalized classification instance. It is defined as:

NormalizedCateName $\sqsubseteq$ $\forall$ partOf NormalizedClasInst
NormalizedCateName $\sqsubseteq$ $\exists$ partOf NormalizedClasInst

ContextualizedTerm

This class represents a contextualized term. That is a given term and the list of classification instances where this term appears as a category name.

Term

This class represents a term. A term corresponds to a normalized category name which has been extracted from the classification system. A term is, at least and only, part of a contextualized term. It is defined as:

Term $\sqsubseteq$ $\forall$ partOf ContextualizedTerm
Term $\sqsubseteq$ $\exists$ partOf ContextualizedTerm

ContextList

This class represents the context of a given term. The context is defined as a list of classification instances where the term was used as a Category Name. The list is defined by specifying the hasContents object property as classification instances, and the isFollowedBy as another ContextList. In addition, a context list is, at least and only, part of a contextualized term. It is defined as:

ContextList $\sqsubseteq$ OWLList

ContextList $\sqsubseteq \forall$ hasContents ClassificationInst
ContextList $\sqsubseteq \forall$ isFollowedBy ContextList
ContextList $\sqsubseteq \exists$ partOf ContextualizedTerm
ContextList $\sqsubseteq \forall$ partOf ContextualizedTerm

TransformationInstance

This class represents the different ways that a normalized classification instance can be turn into during the preprocessing activity. According to the defined transformation at execution time this class semantics has to be defined by meas of its association with existing or new classes.

A.2 Object properties

hasContents

This object property defines the content of each node in the list. Its semantics is defined in the list.owl ontology.

isFollowedBy

This object property defines the class of the next node in the list. Its semantics is defined in the list.owl ontology.

partOf

This object property defines a part of relation. Its inverse relation is *hasPart*. It is defined as:

partOf $\equiv$ hasPart^{-}

hasPart

This object property defines a has part relation. Its inverse relation is *partOf*. It is defined as:

partOf $\equiv$ hasPart^{-}

hasUser

This object property defines the relation has user which is stated always between a classification instance (domain) and a user (range). It is defined as:

$\exists$ hasUser Thing $\sqsubseteq$ ClassificationInst
$\top \sqsubseteq \forall$ hasUser User

hasCategoryName

This object property defines the relation has category name which is stated always between a classification instance (domain) and a category name (range). It is defined as:

$\exists$ hasCategoryName Thing $\sqsubseteq$ ClassificationInst
$\top \sqsubseteq \forall$ hasCategoryName CategoryName

hasClassifiedResource

This object property defines the relation has classified resource which is stated always between a classification instance (domain) and a classified resource(range). It is defined as:

$\exists$ hasClassifiedResource Thing $\sqsubseteq$ ClassificationInst
$\top \sqsubseteq \forall$ hasClassifiedResource ClassifiedResource

hasNormalizedCateName

This object property defines the relation has normalized category name which is stated always between a normalized classification instance (domain) and a normalized category name (range). It is defined as:

$\exists$ hasNormalizedCateName Thing $\sqsubseteq$ NormalizedClasInst
$\top \sqsubseteq \forall$ hasNormalizedCateName NormalizedCateName

hasTerm

This object property defines the relation has term which is stated always between a contextualized term (domain) and a term (range). It is defined as:

$\exists$ hasTerm Thing $\sqsubseteq$ ContextualizedTerm
$\top \sqsubseteq \forall$ hasTerm Term

hasTermContext

This object property defines the relation has context term which is stated always between a contextualized term (domain) and a context list (range). It is defined as:

$\exists$ hasTermContext Thing $\sqsubseteq$ ContextualizedTerm
$\top \sqsubseteq \forall$ hasTermContext ContextList

BIBLIOGRAPHY

Karl Aberer, Philippe C. Mauroux, Aris M. Ouksel, Tiziana Catarci, Mohand S. Hacid, Arantza Illarramendi, Vipul Kashyap, Massimo Mecella, Eduardo Mena, Erich J. Neuhold, and Et. Emergent Semantics Principles and Issues. In *Database Systems for Advances Applications (DASFAA 2004), Proceedings*, pages 25–38. Springer, March 2004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.6378. 14, 70

Sofia Angeletou, Marta Sabou, and Enrico Motta. Semantically enriching folksonomies with FLOR. In *In Proc of the 5th ESWC. workshop: Collective Intelligence amp; the Semantic Web*, 2008. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.141.2569. xvii, 3, 4, 19, 23, 24, 39, 41, 54, 58

Sofia Angeletou, Marta Sabou, and Enrico Motta. Improving folksonomies using formal knowledge: A case study on search. In Asunción Gómez-Pérez, Yong Yu, and Ying Ding, editors, *The Semantic Web*, volume 5926 of *Lecture Notes in Computer Science*, pages 276–290. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-10870-9. URL http://dx.doi.org/10.1007/978-3-642-10871-6_19. 10.1007/978-3-642-10871-6_19. 72

Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search, Second edition.* Pearson Education Ltd., Harlow, England, 2011. ISBN 978-0-321-41691-9. 6, 123, 136, 137

Grigory Begelman, Philip Keller, and Frank Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Proceedings of the WWW Collaborative Web Tagging Workshop*, Edinburgh, Scotland, 2006. 4, 29, 37, 38, 57, 58, 59

Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge. In *Proceedings* *of the 2nd Web Science Conference (WebSci10)*, Raleigh, NC, USA, 2010. 29, 37, 38, 55, 58

Dominik Benz, Christian Körner, Andreas Hotho, Gerd Stumme, and Markus Strohmaier. One tag to bind them all: measuring term abstractness in social metadata. In *Proceedings of the 8th extended semantic web conference on The semanic web: research and applications - Volume Part II*, ESWC'11, pages 360–374, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21063-1. URL `http://dl.acm.org/citation.cfm?id=2017936.2017966`. 28

Tim Berners-Lee. Linked data. *W3C Design Issues*, 2006. URL `http://www.w3.org/DesignIssues/LinkedData.html`. 4

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001. URL `http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21`. 189

Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009a. 4, 5, 101, 136, 189

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. *Journal of Web Semantic*, 7(3):154–165, 2009b. 5, 66, 101, 142, 168, 178

David M. Blei, Andrew Y. Ng, and Michale I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003. URL `http://www.cs.berkeley.edu/~blei/papers/blei03a.ps.gz`. 174, 176

John G. Breslin, Stefan Decker, Andreas Harth, and Uldis Bojars. Sioc&#58; an approach to connect web&#45;based communities. *Int. J. Web Based Communities*, 2:133–142, July 2006. ISSN 1477-8394. doi: 10.1504/IJWBC.2006.010305. URL `http://dl.acm.org/citation.cfm?id=1358978.1358979`. 19

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*, 1998. URL `http://ilpubs.stanford.edu:8090/361/`. 26

Klaus A. Brunner. What's emergent in emergent computing? In Robert Trappl, editor, *Cybernetics and Systems 2002: Proceedings of the 16th European Meeting* *on Cybernetics and Systems Research*, volume 1, pages 189–192, 2002. ISBN 3-85206-160-1. 14

Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. *Ontology Learning from Text: An Overview*, chapter -. IOS Press, 7 2005. 4

Iván Cantador, Martin Szomszor, Harith Alani, Miriam Fernández, and Pablo Castells. Enriching ontological user profiles with tagging history for multi-domain recommendations. In *1st International Workshop on Collective Semantics: Collective Intelligence & the Semantic Web (CISWeb 2008)*, June 2008. URL `http://eprints.ecs.soton.ac.uk/15451/`. xvii, 3, 4, 39, 42, 43, 58

Ciro Cattuto, Dominik Benz, Andreas Hotho, and Gerd 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. ISBN 978-3-540-88563-4. doi: 10.1007/978-3-540-88564-1_39. URL `http://www.kde.cs.uni-kassel.de/pub/pdf/cattuto2008semantica.pdf`. 4, 25, 26, 27, 28, 36, 59

Donald D. Chamberlin and Raymond F. Boyce. SEQUEL: A structured english query language. In *ACM SIGMOD*, page 249Ű264, 5 1974. URL `http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf`. 124

Rudi Cilibrasi and Paul M. B. Vitanyi. Automatic meaning discovery using google, December 2004. URL `http://arxiv.org/abs/cs.CL/0412098`. 160

James Clark. XSL transformations (XSLT) version 1.0. http://www.w3.org/TR/xslt, 1999. 124

William W. Cohen, Pradeep D. Ravikumar, and Stephen E. Fienberg. *A Comparison of String Distance Metrics for Name-Matching Tasks*, pages 73–78. 2003. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.3605`. 126

Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. ISBN 0070131511. 92

Fabio Crestani. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11:453–482, 1997. 6, 7, 66, 97, 123, 130

Chris J. Date. *Database in Depth: Relational Theory for Practitioners: The Relational Model for Practitioners*. O'Reilly Media, 1 edition, 2005. ISBN 0596100124. 92

Gordana Dodig-crnkovic. Scientific methods in computer science. *Computer*, pages 126–130, 2002. URL http://www.mrtc.mdh.se/~gdc/work/cs_method.pdf. 73

Nicholas Drummond, Alan Rector, Robert Stevens, Georgina Moulton, Matthew Horridge, Hai Wang, and Julian Sedenberg. Putting owl in order: Patterns for sequences in owl. In *Workshop OWL Experiences and Directions (OWLEd 2006)*, Athens Georgia, 2006. 196

Francisco Echarte, José Javier Astrain, Alberto Córdoba, and Jesús E. Villadangos. Ontology of folksonomy: A new modelling method. In Siegfried Handschuh, Nigel Collier, Tudor Groza, Rose Dieng, Michael Sintek, and Anita de Waard, editors, *SAAKM*, volume 289 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007. URL http://dblp.uni-trier.de/db/conf/kcap/saakm2007.html#EcharteACV07. 19

Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007. ISBN 3-540-49611-4. 174

Christiane Fellbaum. *WordNet and wordnets*, pages 665–670. Elsevier, Oxford, second edition edition, 2005. URL http://wordnet.princeton.edu/. 91, 104, 175, 177

Mariano Fernández-Lopez, Asunción Gómez-Pérez, and Natalia Juristo. Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium*, pages 33–40, Stanford, USA, March 1997. 60, 117, 118

Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378 – 382, 1971. ISSN 0033-2909. doi: DOI:10.1037/h0031619. URL http://www.sciencedirect.com/science/article/B6WY5-4NRKJ7R-6/2/45574cb4dd27bb7f514b8b5b619f9ec4. 6, 172

Joseph L. Fleiss and Jacob Cohen. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and Psychological Measurement*, 33(3):613–619, 1973. URL http://epm.sagepub.com/cgi/doi/10.1177/001316447303300309. 162

Andrés García-Silva, Oscar Corcho, Harith Alani, and Asunción Gómez-Pérez. Review of the state of the art: discovering and associating semantics to tags in folksonomies. *The Knowledge Engineering Review*, 27(01):57–85, 2012. doi: 10.1017/S026988891100018X. URL http://dx.doi.org/10.1017/S026988891100018X. xvii, 22

Andrés García-Silva, Martin Szomszor, Harith Alani, and Oscar Corcho. Preliminary results in tag disambiguation using dbpedia. In *Knowledge Capture (K-Cap'09) - First International Workshop on Collective Knowledge Capturing and Representation - CKCaR'09*, September 2009. URL `http://oa.upm.es/6377/`. 39, 178

Eirini Giannakidou, Vassiliki Koutsonikola, Athena Vakali, and Yiannis Kompatsiaris. Co-clustering tags and social data sources. In *Web-Age Information Management, 2008. WAIM '08. The Ninth International Conference on*, pages 317 –324, july 2008. doi: 10.1109/WAIM.2008.61. 4, 49, 57, 58

Scott A. Golder and Bernardo A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2006. doi: 10.1177/0165551506062337. URL `http://jis.sagepub.com/cgi/content/abstract/32/2/198`. 3, 14, 19, 130

Jeffrey Goldstein. Emergence as a Construct: History and Issues. *Emergence*, 1(1): 49–72, 1999. URL `http://search.ebscohost.com/login.aspx?direct=true&#38;db=buh&#38;AN=3180006&#38;site=bsi-live`. 14

Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web.* Springer Verlag, 2004. xvii, 76, 77

Asunción Gómez-Pérez and David Manzano-Macho. An overview of methods and tools for ontology learning from texts. *Knowl. Eng. Rev.*, 19(3):187–212, September 2004. ISSN 0269-8889. doi: 10.1017/S0269888905000251. URL `http://dx.doi.org/10.1017/S0269888905000251`. 4

Jorge Gracia and Eduardo Mena. Multiontology Semantic Disambiguation in Unstructured Web Contexts. In *Proc. of Workshop on Collective Knowledge Capturing and Representation (CKCaR'09), Redondo Beach, California (USA)*, September 2009. 159

Jorge Gracia, Jochem Liem, Esther Lozano, Oscar Corcho, Michal Trna, Asunción Gómez-Pérez, and Bert Bredeweg. Semantic techniques for enabling knowledge reuse in conceptual modelling. In *Proceedings of the 9th international semantic web conference on The semantic web - Volume Part II*, ISWC'10, pages 82–97, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-17748-4, 978-3-642-17748-4. URL `http://dl.acm.org/citation.cfm?id=1940334.1940341`. 104

Jorge Gracia, Elena Montiel-Ponsoda, Philipp Cimiano, Asunción Gómez-Pérez, Paul Buitelaar, and John McCrae. Challenges for the multilingual web of data. *Web*

Semant., 11:63–71, March 2012. ISSN 1570-8268. doi: 10.1016/j.websem.2011.09.001. URL `http://dx.doi.org/10.1016/j.websem.2011.09.001`. 103

E. Greenwood. *Metodología de la investigación social.* Editorial Paidos, 1973. 76

Tom Gruber. Ontology of Folksonomy: A Mash-up of Apples and Oranges. *International Journal on Semantic Web & Information Systems*, 3(2), 2007. URL `http://tomgruber.org/writing/ontology-of-folksonomy.htm`. 3, 14, 19

Masahiro Hamasaki, Yutaka Matsuo, and T. Nisimura. Ontology extraction using social network, 2007. URL `http://staff.aist.go.jp/masahiro.hamasaki/index-e.html`. 23, 29, 32, 33, 55, 57, 58

Philipp Heim, Steffen Lohmann, and Timo 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. ISBN 978-3-642-13485-2. 66, 123, 141, 178

Paul Heymann and Hector Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford InfoLab, April 2006. URL `http://ilpubs.stanford.edu:8090/775/`. 29, 37, 38, 55, 57, 58

Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. *Information Retrieval in Folksonomies: Search and Ranking*, volume 4011 of *Lecture Notes in Computer Science*, chapter 31, pages 411–426. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-34544-2. doi: 10.1007/11762256_31. URL `http://www.kde.cs.uni-kassel.de/stumme/papers/2006/hotho2006information.pdf`. 2

IEEE. Standard glossary of data management terminology. *IEEE Std 610.5-1990*, page 1, 1990a. doi: 10.1109/IEEESTD.1990.94601. 75

IEEE. Standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, page 1, 1990b. doi: 10.1109/IEEESTD.1990.101064. 76

Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901. 25

Robert Jäschke, Andreas Hotho, Christoph Schmitz, Bernhard Ganter, and Stum Gerd. Discovering shared conceptualizations in folksonomies. *Web Semant.*, 6:38–53, February 2008. ISSN 1570-8268. doi: 10.1016/j.websem.2007.11.004. URL http://dl.acm.org/citation.cfm?id=1346366.1346701. 4, 29, 33, 34, 57, 58

Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, cmp-lg/9709008, 1997. URL http://dblp.uni-trier.de/db/journals/corr/corr9709.html#cmp-lg-9709008. 177, 178

Matjaz B. Juric. *Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition*. Packt Publishing, 2006. ISBN 1904811817. 77

Martin Kavalec and Vojtech Svátek. Information extraction and ontology learning guided by web directory. In *ECAI Workshop on NLP and ML for ontology engineering. Lyon*, 2002. 15

Lyndon Kennedy, Mor Naaman, Shane Ahern, Rahul Nair, and Tye Rattenbury. How flickr helps us make sense of the world: context and content in community-contributed media collections. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 631–640, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-702-5. doi: http://doi.acm.org/10.1145/1291233.1291384. URL http://doi.acm.org/10.1145/1291233.1291384. 23, 29, 37, 38, 39, 57, 58

Hak-Lae Kim, John G. Breslin, Sung-Kwon Yang, and Hong-Gee Kim. Social semantic cloud of tag: semantic model for social tagging. In *Proceedings of the 2nd KES International conference on Agent and multi-agent systems: technologies and applications*, KES-AMSTA'08, pages 83–92, Berlin, Heidelberg, 2008a. Springer-Verlag. ISBN 3-540-78581-7, 978-3-540-78581-1. URL http://dl.acm.org/citation.cfm?id=1787839.1787849. xvii, 19, 20

Hak Lae Kim, Simon Scerri, John G. Breslin, Stefan Decker, and Hong Gee Kim. The state of the art in tag ontologies: a semantic model for tagging and folksonomies. In *Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications*, pages 128–137. Dublin Core Metadata Initiative, 2008b. URL http://dl.acm.org/citation.cfm?id=1503418.1503431. 19, 72

Ralph Kimball and Joe Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleanin*. John Wiley & Sons, 2004. ISBN 0764567578. 87, 93

Thomas Knerr. Tagging ontology - towards a common ontology for folksonomies, 2006. URL http://code.google.com/p/tagont/. 19

Christian Körner, Dominik Benz, Andreas Hotho, Markus Strohmaier, and Gerd Stumme. Stop thinking, start tagging: tag semantics emerge from collaborative verbosity. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 521–530, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772744. URL http://doi.acm.org/10.1145/1772690.1772744. 27

C. R. Kothari. *Research methodology: Methods and techniques.* New age international publishers., second edition edition, 2004. 72, 73

Yannis Labrou and Tim Finin. Yahoo! as an ontology: using yahoo! categories to describe documents. In *Proceedings of the eighth international conference on Information and knowledge management*, CIKM '99, pages 180–187, New York, NY, USA, 1999. ACM. ISBN 1-58113-146-1. doi: 10.1145/319950.319976. URL http://doi.acm.org/10.1145/319950.319976. 15

Sun-Sook Lee and Hwan-Seung Yong. Tagplus: A retrieval system using synonym tag in folksonomy. In *Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on*, pages 294–298. IEEE, 2007. 19

Freddy Limpens, Fabien Gandon, L., and Michel Buffa. Helping online communities to semantically enrich folksonomies. In *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line, http://webscience.org*, pages 1–8, Raleigh, États-Unis, 2010. URL http://hal.archives-ouvertes.fr/hal-00530364. 36, 55, 57, 58

Hugo Liu and Pattie Maes. Interestmap: Harvesting social network profiles for recommendations. In *Beyond Personalisation*, December 2004. URL http://web.media.mit.edu/~hugo/publications/papers/BP2005-hugo-interestmap.pdf. 15

Mohamed Zied Maala, Alexandre Delteil, and Ahmed Azough. A conversion process from Flickr tags to RDF descriptions. *IADIS INTERNATIONAL JOURNAL ON WWW/INTERNET*, 6(1), 2008. ISSN 1645-7641. URL http://liris.cnrs.fr/publis/?id=4425. 39, 47, 57, 58

Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *Intelligent Systems, IEEE*, 16(2):72 – 79, mar-apr 2001. ISSN 1541-1672. doi: 10.1109/5254.920602. 4

Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominik Benz, Andreas Hotho, and Stum Gerd. Evaluating similarity measures for emergent semantics of social tagging. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 641–650, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: http://doi.acm.org/10.1145/1526709.1526796. URL `http://doi.acm.org/10.1145/1526709.1526796`. 4, 25, 26

Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. Ht06, tagging paper, taxonomy, flickr, academic article, to read. In *Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 31–40. ACM, 2006. 3, 14, 59

Paul Martin and Gagarine Yaikhom. *DISPEL Reference Manual.* 2011. URL `www.admire-project.eu`. xvii, 5, 67, 77, 80

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics)*, 2011. 104

Peter Mika. Ontologies are us: A unified model of social networks and semantics. *Web Semant.*, 5:5–15, March 2007. ISSN 1570-8268. doi: 10.1016/j.websem.2006.11.002. URL `http://dl.acm.org/citation.cfm?id=1229184.1229195`. xvii, 3, 4, 23, 25, 29, 30, 37, 57, 58, 70, 130

Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33:31–88, March 2001. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/375360.375365. URL `http://doi.acm.org/10.1145/375360.375365`. 125

Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2): 10:1–10:69, February 2009. ISSN 0360-0300. doi: 10.1145/1459352.1459355. URL `http://doi.acm.org/10.1145/1459352.1459355`. 22, 23

Richard Newman. Tag ontology design, December 2005. URL `http://www.holygoat.co.uk/projects/tags/`. 19

Alexandre Passant. Using Ontologies to Strengthen Folksonomies and Enrich Information Retrieval in Weblogs. In *Proceedings of the First International Conference on Weblogs and Social Media (ICWSM)*, Boulder, Colorado, March 2007. URL `http://www.icwsm.org/papers/paper15.html`. 39, 46, 55, 58

Alexandre Passant and Philippe Laublet. Meaning of a tag: A collaborative approach to bridge the gap between tagging and linked data. *Proceedings of the WWW*

2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China, Apr, 2008. URL `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-369/paper22.pdf`. 19

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet::similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, pages 38–41, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1614025.1614037`. 177

Helena Sofia Pinto, Christoph Tempich, and Steffen Staab. DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engingeering of oNTologies. In Ramon Lopez de Mantaras and Lorenza Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 393–397. IOS Press, August 2004. 61

Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF, 2008. URL `http://www.w3.org/TR/rdf-sparql-query/`. 6, 66, 123

Dorian Pyle. *Data Preparation for Data Mining (The Morgan Kaufmann Series in Data Management Systems).* Morgan Kaufmann, March 1999. ISBN 1558605290. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1558605290`. 84

Leonard Richardson and Sam Ruby. *Restful web services.* O'Reilly, first edition, 2007. ISBN 9780596529260. 124

Heather J. Ruskin and Ray Walshe. Emergent Computing - Introduction to the Special Theme. *ERCIM News*, (63):24–25, 2006. 14

Gerard Salton and Michael J. Mcgill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA, 1986. 6, 8, 66, 104, 123, 125, 137, 174, 175

Simon Scerri, Michael Sintek, Ludger van Elst, and Siegfried Handschuh. NEPOMUK Annotation Ontology Specification, August 2007. URL `http://www.semanticdesktop.org/ontologies/nao/`. 19

Anne Schlicht and Heiner 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. ISBN 978-0-7695-3496-1.

doi: 10.1109/WIIAT.2008.398. URL http://portal.acm.org/citation.cfm?id=1486927.1487100. 171

Christoph Schmitz, Andreas Hotho, Robert Jäschke, and Gerd Stumme. Mining association rules in folksonomies. In V. Batagelj, H.-H. Bock, A. Ferligoj, and A. Žiberna, editors, *Data Science and Classification. Proceedings of the 10th IFCS Conf.*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 261–270, Heidelberg, July 2006. Springer. 33

Claude E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948. 27

Ahu Sieg, Bamshad Mobasher, and Robin Burke. Improving the effectiveness of collaborative recommendation with ontology-based user profiles. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 39–46, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0407-8. doi: 10.1145/1869446.1869452. URL http://doi.acm.org/10.1145/1869446.1869452. 15

Lucia Specia and Enrico Motta. Integrating folksonomies with the semantic web. In E. Franconi, M. Kifer, and W. May, editors, *The Semantic Web: Research and Applications*, pages 624–639. Springer, Berlin, 2007. URL http://oro.open.ac.uk/15676/. xvii, 3, 23, 49, 51, 53, 55, 58, 70

Steffen Staab, Alexander Mädche, Frank Nack, S. Santini, and Luc Steels. Emergent semantics. *IEEE Intelligent Systems*, 17(1):78–86, January 2002. Trends & Controversies. 3, 14

Mari Carmen Suarez-Figueroa. *NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse.* PhD thesis, Facultad de Informática (UPM), Madrid, Spain, 2010. 61

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A Large Ontology from Wikipedia and WordNet. *Elsevier Journal of Web Semantics*, 2008. 168

York Sure, Steffen Staab, and Rudi Studer. On-to-knowledge methodology (otkm). In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies: International Handbook on Information Systems*, pages 117–132. Springer, 2004. ISBN 3540408347. 60

Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, and Aldo Gangemi. *Ontology Engineering in a Networked World.* Springer, Berlin, 2012. 4, 65, 77, 117, 118

Martin Szomszor, Harith Alani, Ivan Cantador, Kieron O'Hara, and Nigel Shadbolt. Semantic modelling of user interests based on cross-folksonomy analysis. In *7th International Semantic Web Conference (ISWC)*, October 2008. URL `http://eprints.ecs.soton.ac.uk/16551/`. 19

Don Tapscott and Anthony D. Williams. *Wikinomics: How Mass Collaboration Changes Everything.* Portfolio Hardcover, December 2006. ISBN 1591841380. URL `http://www.worldcat.org/isbn/1591841380`. 1

Maurizio Tesconi, Francesco Ronzano, Andrea Marchetti, and Salvatore Minutoli. Semantify del. icio. us: automatically turn your tags into senses. In *Social Data on the Web, Workshop at the 7th ISWC*, 2008. 3, 4, 39, 44, 57, 58

Vasudeva Varma. Building large scale ontology networks. In *Language Engineering Conference, 2002. Proceedings*, pages 121 – 127, dec. 2002. doi: 10.1109/LEC.2002.1182299. 15

Boris Villazón-Terrazas. *A Method for Reusing and Re-engineering Non-ontological Resources for Building Ontologies.* 2012. 59, 62, 63, 65, 118

Boris Villazón-Terrazas, Mari Carmen. Suárez-Figueroa, and Asunción Gómez-Pérez. A pattern-based method for re-engineering non-ontological resources into ontologies. *International Jounal on Semantic Web and Information Systems*, 6(4):27–63, 2010. URL `http://www.igi-global.com/bookstore/article.aspx?titleid=50498`. 77

Stanley Wasserman, Katherine Faust, Dawn Iacobucci, and Mark Granovetter. *Social Network Analysis: Methods and Applications.* Cambridge University Press, 1994. 23

Tom White. Did you mean: Lucene?, 2005. URL `http://today.java.net/pub/a/today/2005/08/09/didyoumean.html`. 125

Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/981732.981751. URL `http://dx.doi.org/10.3115/981732.981751`. 40, 107, 177

Ching Man Au Yeung, Nicholas Gibbins, and Nigel Shadbolt. Understanding the semantics of ambiguous tags in folksonomies. In *The International Workshop on Emergent Semantics and Ontology Evolution (ESOE2007) at ISWC/ASWC 2007*, November 2007. URL `http://eprints.ecs.soton.ac.uk/14869/`. 23

Justin Zobel and Philip Dart. Finding approximate matches in large lexicons. *Software - Practice & Experience*, 25(3):331–345, March 1995. 121, 125