PhD Dissertation

International Doctorate School in
Information and Communication Technologies

DISI - University of Trento

Universidad Católica
"Nuestra Señora de la Asunción"

DISTRIBUTED IDENTITY MANAGEMENT

Juan Ignacio Pane Fernández

Advisor:
Prof. Fausto Giunchiglia
University of Trento

Co-Advisor:
Prof. Luca Cernuzzi
Universidad Católica "Nuestra Señora de la Asunción"

March 2012

Abstract

Semantics is a local and a global problem at the same time. Local because is in the mind of the people who have personal interpretations, and global because we need to reach a common understanding by sharing and aligning these personal interpretations.

As opposed to current state-of-the-art approaches based on a two layer architecture (local and global), we deal with this problem by designing a general three layer architecture rooted on the personal, social, and universal levels. The new intermediate social level acts as a global level for the personal level, where semantics is managed around communities focusing on specific domains, and as local for the universal level as it only deals with one part of universal knowledge.

For any of these layers there are three main components of knowledge that helps us encode the semantics at the right granularity. These are: *i)* Concrete knowledge, which allows us to achieve semantic compatibility at the level of entities, the things we want to talk about; *ii)* Schematic knowledge, which defines the structure and methods of the entities; and *iii)* Background knowledge, which enables compatibility at the language level used to describe and structure entities.

The contribution of this work is threefold: *i)* the definition of general architecture for managing semantics of entities, *ii)* the development components of the system based on the architecture; these are structure preserving semantic matching and sense induction algorithms, and *iii)* the evaluation of these components with the creation of new gold standards datasets.

To my wife Gloria, my son Giorgio, and my parents Jorge and Estela, whose constant love and support helped me achieve this goal and many more. I am truly blessed to have them in my life.

Acknowledgment

I would like to thank Prof. Luca Cernuzzi for believing in me and opening the door to the opportunity of doing the PhD at Trento.

This PhD would also have not been possible without the guidance and constant support of my advisor, Prof. Fausto Giunchiglia, from whom I have learned so much and I hope to continue learning.

I would also like to thank Ilya Zahirayeu, Pavel Shvaiko and Pierre Andrews, who dedicated so much of their time to teach me how to do research when I needed it the most. With help of each of them I was able to develop a part of this thesis, and for that, I will be eternally grateful.

Also, I would like to thank all the members of the Knowdive group, with whom I have had, and continue to have many fruitful conversations and feedback on a daily basis. There are many other people that have helped come this far, to all of you, thank you very much!

The work presented in Chapter 5: "Semantic Matching" and Chapter 10: "Evaluation of Structure Preserving Semantic Matching" was carried out with the contribution of Mikalai Yatskevich, Pavel Shvaiko, Lorenzo Vaccari, Paolo Besana and Fiona Mc'Neill and is partially published in [GMY+08], [MBPG09], [VSP+12] and [GAP12] in the scope of the EU funded project OpenKnowledge `http://www.openk.org` (FP6-027253) .

The work presented in Chapter 3: "Knowledge Components", Chapter 6: "Background Knowledge Evolution" and Chapter 11: "Evaluation of Background Knowledge Evolution" was carried out with the contribution of Ilya Zaihrayeu and Pierre Andrews and is partially published in [APZ11a], [AZP12] and [APZ11b] in the scope of the EU funded project Insemtives `http://www.insemtives.eu` (FP7-231181).

Finally, **to my wife Gloria**, for letting me start this chapter of my life that ends with this book, and for being always at my side with constant love and support. None of this would have been possible without her.

Juan Pane: List of publications

[AKPZ11] Pierre Andrews, Sergey Kanshin, Juan Pane, and Ilya Zaihrayeu, *Semantic Annotation of Images on Flickr*, Proceedings of ESWC 2011 (Grigoris Antoniou, Marko Grobelnik, and Elena Simperl, eds.), LNCS, vol. 6644, Springer, June 2011, pp. 476–480.

[APZ11a] Pierre Andrews, Juan Pane, and Ilya Zaihrayeu, *Semantic disambiguation in folksonomy: A case study*, Advanced Language Technologies for Digital Libraries (Raffaella Bernardi, Sally Chambers, Bjrn Gottfried, Frdrique Segond, and Ilya Zaihrayeu, eds.), Lecture Notes in Computer Science, vol. 6699, Springer Berlin / Heidelberg, 2011, pp. 114–134.

[APZ11b] Pierre Andrews, Juan Pane, and Ilya Zaihrayeu, *Sense induction in folksonomies*, Workshop on Discovering Meaning On the Go in Large Heterogeneous Data 2011 (Fiona McNeill, Harry Halpin, and Michael Chan, eds.), International Joint Conference on Artificial Intelligence, 2011.

[APZA11] Pierre Andrews, Juan Pane, Ilya Zaihrayeu, and Aliaksandr Autayeu, *Supporting semantic annotations in flickr*, Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on, aug. 2011, pp. 187 –194.

[AZP12] Pierre Andrews, Ilya Zaihrayeu, and Juan Pane, *A classification of semantic annotation systems*, Semantic Web Journal (2012), no. Interoperability, Usability, Applicability, to appear.

[CEH+08] Caterina Caracciolo, Jérôme Euzenat, Laura Hollink, Ryutaro Ichise, Antoine Isaac, Véronique Malaisé, Christian Meilicke, Juan Pane, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, and Vojtech Svátek, *Results of the ontology alignment evaluation initiative 2008*, Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008) Collocated with the 7th International Semantic Web Conference (ISWC-2008) (Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, eds.), CEUR Workshop Proceedings, vol. 431, CEUR-WS.org, 2008.

[EFH+09] Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, François Scharffe, Pavel Shvaiko, Vassilis Spiliopoulos, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, Cássia Trojahn dos Santos, George A. Vouros, and Shenghui Wang, *Results of the ontology alignment evaluation initiative 2009*, Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009) (Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Natalya Fridman Noy, and Arnon Rosenthal, eds.), CEUR Workshop Proceedings, vol. 551, CEUR-WS.org, 2009.

[EFM+10] Jérôme Euzenat, Alfio Ferrara, Christian Meilicke, Juan Pane, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, and Cássia Trojahn dos Santos, *Results of the ontology alignment evaluation initiative 2010*, Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) (Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Ming Mao, and Isabel F. Cruz, eds.), CEUR Workshop Proceedings, vol. 689, CEUR-WS.org, 2010.

[GAP12] Fausto Giunchiglia, Aliaksandr Autayeu, and Juan Pane, *S-Match: an open source framework for matching lightweight ontologies*, Semantic Web Journal (2012), no. Semantic Web Tools and Systems, to appear.

[GMY+08] Fausto Giunchiglia, Fiona McNeill, Mikalai Yatskevich, Juan Pane, Paolo Besana, and Pavel Shvaiko, *Approximate structure-preserving semantic matching*, Proceedings of the OTM 2008 Confederated International Conferences. ODBASE 2008 (Moterrey, Mexico), Springer-Verlag, 2008, pp. 1217–1234.

[MBPG09] Fiona McNeill, Paolo Besana, Juan Pane, and Fausto Giunchiglia, *Service integration through structure preserving semantic matching*, Cases on Semantic Interoperability for Information Systems Integration:Practices and Applications **11** (2009), no. Semantic Interoperability, 26–45.

[MSVP08] Maurizio Marchese, Pavel Shvaiko, Lorenzino Vaccari, and Juan Pane, *An application of approximate ontology matching in eResponse*, 5th International ISCRAM conference (Washington. USA), 2008.

[VSP+12] Lorenzino Vaccari, Pavel Shvaiko, Juan Pane, Paolo Besana, and Maurizio Marchese, *An evaluation of ontology matching in geo-service applications*, GeoInformatica (Journal) **16** (2012), 31–66.

[ZPdVA11] Ilya Zaihrayeu, Juan Pane, Germán Toro del Valle, and Pierre Andrews, *Adopting semantic annotation systems for corporate portal management: a telefonica case study*, Proceedings of the 7th International Conference on Semantic Systems (New York, NY, USA), I-Semantics '11, ACM, 2011, pp. 179–182.

Contents

List of Figures

List of Tables

Part I

General notions

Chapter 1

Introduction

In the Web 2.0 era knowledge made available in digital format is growing at a fast pace. Individuals, communities and organizations usually have their own local models and standards to represent their knowledge, which evolves over time, creating semantically heterogeneous knowledge bases, even if dealing with the same domain. Given the proliferation of information systems, and the growing specialization in key areas (e.g., bioinformatics, e-government, e-health), there is a need to collaborate in order to build shared innovative solutions, which would benefit from the knowledge that can be obtained by combining information from different sources. This is especially true in domains facing identification problems, for example, in governmental agencies that need to gather information from different sources for fighting tax evasion, or for health researches that need to combine patient data to measure, for example, the effect of a treatment. This collaborative process poses the need of bridging the interoperability issues coming from heterogeneous knowledge representations.

The aim of this thesis is to propose a framework for dealing with the semantic heterogeneity issue in distributed scenarios in order to maximize interoperability. We organize our knowledge framework around entities, i.e., things we want to represent, describe and manage. There are several sources of semantic heterogeneity such as the natural language used to describe the entities and their schemas and the set of attributes chosen to describe the entities. These need to be resolved in order to achieve interoperability and to enable collaboration between the participants of the distributed scenario.

We adopt an a-priori approach for dealing with semantic heterogeneity. This is, instead of the classical post-hoc approach (schema matching, ontology matching, record linkage, ...) we tackle the problem by defining a Knowledge Organization System (KOS) that deals with the problem by construction. We propose a KOS model centered around entities where the knowledge is defined using three main components: *i)* the *background knowledge* that deals with the ambiguity of natural language by converting natural language labels into a formal language (language independent concepts) that is machine processable; *ii)* the *schematic knowledge*

that deals with the heterogeneity at schema level, allowing us to define different entity types based on the formal language of the background knowledge; and *iii)* the *concrete knowledge* that contains the model of the entities, based on the entity types defined in the schematic knowledge and where the values for the attributes of the entities can be expressed as references to other entities, as concepts from the formal language, besides the classical set of basic data types such as integer, float, date, etc.

Capturing the semantics of the entities in this distributed scenario is local and a global problem at the same time. Local because the entities are to be provided by people who have personal interpretations and that would want to provide descriptions about their entities using their personal (local) knowledge (terms, concepts, structure of the entities); and global because in order to profit from the distributed scenario, where people benefit from the knowledge created by others, we need to reach a common understanding by sharing and aligning these personal (local) interpretations.

We propose to achieve interoperability by capturing the real world semantics of knowledge systems created at three different layers: local (personal), social (communities) and global (across communities). Where in the first layer people create and maintain locally their own vision of the world, then they share this knowledge with other people and interact in several bounded social contexts (their communities), but where in order to achieve successful multi-context interaction there is a need of an upper layer that formalizes what is commonly known across communities. We believe that in such scenario entities will be created mostly in a bottom-up fashion, and that interoperability will be achieve by reusing the background (language and concepts) and schematic (entity types) knowledge mainly in a top-down fashion.

In this distributed scenario we need to also deal with the dynamic nature of knowledge, this is, while we can enable interoperability by reusing a predefined set of words and concepts in a top-down fashion, we also need to provided the capabilities for extending such set of words and concepts based on their usage by the participants of the distributed scenario. This is needed because relying on a fixed vocabulary can hinder the capability of the system to extract machine processable semantics from natural language labels. As we will see, this is a real problem in highly dynamic scenarios such as Computer Science. Once the system is capable of creation new terms and concepts, either automatically by relying on the process described before, or manually by the user, we also need to define the procedure by which new knowledge is propagated among the participants of the distributed system.

Given that entities are to be created and maintained locally by people, providing different perspectives about the same entity being referred, we need to deal with incomplete and inconsistent perspectives. The aim of this thesis is to deal with the heterogeneity by being able to unequivocally and globally identify the entities being referred under such conditions. This poses the need of first understanding the types of entities that come into play, and how to identify each of them. We

identify three types of entities: *i)* real world entities, i.e., those that actually exits; *ii)* mental entities, i.e., those that people from in their mind from their own perspective referring to real world entities based on their experience and the interaction with the world; and *iii)* digital entities, i.e., those that are encoded in the system as provided by people (based on their mental entities) referring to real wold entities. Under such condition, the digital entity should be able to distinguish between the identity of the actual real world entity, and the identity of the local perspective encoded in the system. This difference will prove to be key in the design of the system, and will allows us to propose a solution capable of achieving the goal of capturing the real world semantics of the real world entities, given partial views about them.

The contribution of this thesis is threefold:

1. the definition of a general architecture for managing and enabling interoperability of entities in a distributed setting, rooted in the notion of the identification of entities;

2. the development of components of the system based on the defined architecture, these are:

   - S-Match, an open source framework for semantic matching (`http://www.s-match.org`).
   - the Tags2Con "Delicious" dataset is, to the best of our knowledge, the first publicly available dataset included in the Linked Open Data initiative that contains manually created links from Delicious tags to Wornet synsets for disambiguating the meaning of the tags (`http://disi.unitn.it/~knowdive/dataset/delicious/`).
   - the development of a generic sense induction algorithm that can be extended with new distance metrics.

3. the evaluation of the above mentioned components.

1.1 Thesis organization

The structure of the thesis is as follows:

In part I: *General notions* introduces the problem statement that is to be addressed in this thesis.

Part II: *Global architecture for managing local semantics* is composed by the following chapters:

- Chapter 3: "Knowledge Components" presents the structure of the Knowledge Organization System that is designed to deal with different sources semantic heterogeneity. The *background knowledge* is composed of a linguistic part defining the natural languages that are to be supported by the system linked to a conceptual part that is unambiguous enabling automatic reasoning capabilities; the *schematic knowledge* is composed by the entity

type definitions and the *concrete knowledge* contains the entities of the system;

- Chapter 4: "The semantic layers" outlines the three layers of knowledge: Universal, Community and Personal that will allow people to communicate with each other in a fully distributed manner, but also in bounded contexts, while enabling interoperability;

Part III: *Background and Schema compatibility* deals with the issue of how to achieve interoperability at the level of language and schema and is composed by the following chapters:

- Chapter 5: "Semantic Matching" presents an algorithm that allows the system to extract formal semantics from natural language labels and to use this to compare tree-like such as schemas or entity types;

- Chapter 6: "Background Knowledge Evolution" presents a sense induction algorithm for the computation of new concepts based on the usage of terms in a distributed setting, and the addition of these new concepts to the linguistic and conceptual part of the background knowledge;

- Chapter 7: "Knowledge propagation" presents a pragmatic approach for the propagation of knowledge from the different components of the knowledge presented in Chapter 3 between all the layers presented in Chapter 4;

Part IV: *Entity compatibility* studies how to achieve interoperability at the level of entities and is composed by the following chapters:

- Chapter 8: "Entity Compatibility" defines the conditions under which, given the incomplete and inconsistent perspectives about entities, it can be established whether two perspectives refer to the same real world entity or not;

- Chapter 9: "Identity Management" defines how to manage the identity of the real world entity being referred by different perspectives by different people, even using different terminologies. The core contribution of this thesis is described in this Chapter gluing all the previous notions presented in the previous chapters;

Part V: *Evaluation* presents the evaluation of two components: "Semantic Matching" and "Background Knowledge Evolution" and is composed of the following chapters.

- Chapter 10: "Evaluation of Structure Preserving Semantic Matching" evaluates the Structure Preserving Semantic Matching algorithm defined in Chapter 5.

- Chapter 11: "Evaluation of Background Knowledge Evolution" presents the evaluation methodology and initial evaluation of the sense induction algorithms defined in Chapter 6. The contribution of the chapter is also a public gold standard dataset based on Delicious containing links to Wordnet. The dataset is used for the evaluation of the sense induction algorithm and can also be used to evaluate Natural Language Processing (NLP) and Word Sense Disambiguation (WSD) algorithms.

Part VI: *Concluding Remarks* present the conclusions of the work and highlights the contributions. The Future Work Chapter outlines the issues still open to research that will enable a system to fully address the issue of identity management in distributed scenarios.

Chapter 2

Problem statement

The goal of the thesis is to define a distributed system that allows people to manages their knowledge locally. People would normally want to manage things they are interested in according to their personal point of views but being able to effectively collaborate with each other. These *things* people care about need to be properly modeled to achieve interoperability between people's systems; we will model these *things* as entities.

The problem, in such distributed scenario, *is how to capture the real world semantics of entities in order to be able to manage their identity to achieve interoperability.*

This problem, is composed by several subproblems, namely:

1. In a distributed setting, there are several sources of heterogeneity, the language, the schema, the ways entities are described, the different perspectives from which they are described, and the identifiers used.

2. Semantics is a local and a global problem at the same time. Local because the entities are created locally by people, and global because there is a need of shared understanding if we are to allow people to effectively collaborate.

3. People use natural language in order to communicate with each other. However, natural language is inherently ambiguous and the computer does not have the same capability humans have to understand this language and exploit its semantics. We need to formalize the natural language so that its semantics can also be exploited by the computer system.

4. If we are to capture the real world semantics of entities in a distributed scenario, the solution cannot operate under a closed world assumption, i.e., knowledge is inherently dynamic in nature, and the solution cannot be a static knowledge system but a system that supports the constant evolution of knowledge.

5. Given this evolution of knowledge, new knowledge has to be propagated between the participants of the system.

6. Finally, entities are created and maintained by people from different perspectives, which means that we need to deal with incomplete and inconsistent perspectives for the same entity being referred.

In this thesis we deal with each of these problems, namely:

- in Chapter 3: "Knowledge Components" we present the structure of the knowledge for dealing with different semantic heterogeneity sources;
- in Chapter 4: "The semantic layers" we outlined three layers of knowledge: Universal, Community and Personal that will allow people to communicate with each other in a fully distributed manner, but also in bounded contexts, while enabling interoperability;
- in Chapter 5: "Semantic Matching" we outline a state-of-the-art algorithm that allows the system to extract formal semantics from natural language labels;
- in Chapter 6: "Background Knowledge Evolution" we present a sense induction algorithm that can compute new concepts based on the usage of terms in a distributed setting;
- in Chapter 7: "Knowledge propagation" we present a pragmatic approach that enables the systems to propagate knowledge from the different components of the knowledge presented in Chapter 3 between all the layers presented in Chapter 4;
- in Chapter 8: "Entity Compatibility" we defined the conditions under which, given the incomplete and inconsistent perspectives about entities, we can establish whether two perspectives refer to the same real world entity or not;
- in Chapter 9: "Identity Management" we finalize the research by establishing how to manage the identity of the real world entity being referred by different perspectives by different people, and even using different terminology. The core contribution of this thesis is described in this Chapter gluing all the previous notions presented in the previous chapters;
- finally in Chapter 10: "Evaluation of Structure Preserving Semantic Matching" and Chapter 11: "Evaluation of Background Knowledge Evolution" we evaluate two core components, Semantic Matching and Background Knowledge Evolution, tackling problems 3 and 4 from the previous list.

Part II

Global architecture for managing local semantics

Chapter 3

Knowledge Components

3.1 Introduction

When knowledge and structured information is developed and maintained in distributed scenarios by independent, unrelated organizations, it will most likely have different structures, even when dealing with the same domain. This difference in knowledge organization is known as *the semantic heterogeneity problem*. The difficulties faced when trying to integrate these heterogeneous sources of information is known as *the interoperability problem* [Hal05]. However, there is a need to collaborate in order to build shared information systems, which benefit from the knowledge that can be obtained by combining information from different sources. This collaborative process poses the need of working with heterogeneous knowledge [HM93].

The heterogeneity problem can be dealt with at different levels. The authors in [PH00] identify three main sources of semantic heterogeneity: structural, domain and data. We argue that structural and domain can be merged, and distinguish a more basic level that the authors recognized as a problem at all their levels, the natural language. Therefore, our new approach recognizes the following levels:

1. Language: when naming attributes and values there are several issues such as the language used (e.g., "English" vs. "Italian"), the domain terminology used to refer to the same object or concept, even in the same language (e.g., "dog" vs "Canis lupus familiaris"), the level of specificity used (e.g., "dog" vs "animal"), besides the well known issues such as case sensitivity, synonyms, acronyms, homonyms, and word variations.

2. Schema: refers to the structure given to the data to be managed in the application domain. Even in the same domain there are issues such as the name of the properties, the ordering, the exact choice of properties used, aggregation of properties (e.g., {price+taxes} or {price,taxes}) [PH00].

3. Data: refers to how the specific data instances can be different. For example, different identifiers, partial information, incorrect information and different

Figure 3.1: A depiction of the same object (a person) at different points in time showing how the time dimension is a source for heterogeneity when describing entities.

formats in the values. The time validity is another important source of heterogeneity and possible conflicts. The same object can be described differently according to when the description was created. Figure 3.1 shows an example of how a description of the same person (in this case a picture) can greatly vary according to the time dimension.

We adopt an a-priory approach to dealing with semantic heterogeneity. This is, instead of the classical post-hoc approach, where for each of the sources of heterogeneity there is already a well established community researching how to integrate existing heterogeneous sources (schema matching, data integration, ontology alignment) we want to tackle the heterogeneity problem by defining a Knowledge Organization System (KOS) that deals with the above mentioned heterogeneity problems by design.

We propose a KOS model centered in the data, which we call entities, how the entities can be structured, i.e., the schema, and how the entities can be described, i.e., the language used. Considering this, we define three main components, each dealing with a specific heterogeneity issue:

1. background knowledge: containing the natural language dictionary in several languages, but given the ambiguity of the natural language, we convert all the natural language into a formal language, i.e., a set of concepts with relations between them. This component already deals with the difference in values due to synonymy, homonymy, variations of words, different language, etc.

2. schematic knowledge: containing a general framework for defining types of entities, using as basic building blocks the concepts in the background

knowledge. This means that when naming an attribute, there is no issue of ambiguous names, where the order of the attributes is not important, because they are all formalized and easily comparable, even when different schemas are used for the same entity type for different application domains.

3. concrete knowledge: the entities (data) we want to manage in the system. We center our attention in capturing the semantics of the real world entities as provided by people, therefore dealing by construction with different points of view. We therefore recognize the difference between the identification of the real world entity and the identification of the, possible different, (partial) views provided by people.

In the remainder of the thesis, a system that follows the above-mentioned architecture will be referred to as a knowledge base.

The rest of the Chapter is organized as follows. In Section 3.2 we present the basic building block of our framework, the linguistic and conceptual parts, explaining why the linguistic part is not enough and how by having the conceptual part, we can already solve much of the semantic heterogeneity issues. In Section 3.3 we present the model for defining entity types and in Section 3.4 we define what we mean by entities, and the difference between real world entities, mental entities (those provided by people about the real world entities) and digital entities (those encoded in the system that captures the semantics of the real world entity as provided by people). In Section 3.5 we compare our approach to the related work. Finally, Section 3.6 summarized the Chapter.

3.2 Background knowledge[1]

Human (and machines) need a common language in order to communicate with each other. Natural language is the means by which humans understand each other; however, as noted by [SRT05, GH06], natural language, is inherently ambiguous semantically, but also syntactically. A number of issues with natural language, also reported in [APZ11a] are:

Base form variation This problem is related to natural language input issues where a description is based on different forms of the same word (e.g., plurals vs. singular forms, conjugations, misspellings) [GH06].

Homography Elements of the language may have ambiguous interpretation. For example, the term "*atm*" may mean, in the physics domain: a standard unit of pressure, "*the atmosphere*"; or in the banking domain: a "*cash machine*";

Synonymy Syntactically different words may have the same meaning. For example, the words "*image*" and "*picture*" may be used interchangeably by people

[1] Parts of this section were published in [APZ11a] in collaboration with Pierre Andrews and Ilya Zaihrayeu.

but will be treated by the system as two different elements of the language because of their different spelling;

Specificity gap This problem comes from a difference in the specificity of words used when describing things. For example, a user might describe a picture with the term "*dog*", while another can describe the same picture with the term "*Labrador*", which is more specific than dog, and the system can treat both terms as not related due to the syntactic difference.

The above mentioned issues make natural language unsuitable for the communication between machines, which do no poses (to the best of our knowledge) the skills we as humans use to overcome them. Therefore, a more formal language is needed for machines to be able to successfully interact. Sheth [SRT05] calls this formal semantics a representation model such that there is a definite semantic interpretation, making them machine processable. An example of a formalism with formal semantics can be found in ontologies.

This work follows the approach outlined in [GMFD10], thus identifying two main components of the background knowledge: i) the linguistic part, containing the natural language elements that make as bridge between what is provided by humans (in natural language) and ii) and conceptual part formalizing what is processable by the machines.

3.2.1 The linguistic part

The linguistic part follows the model introduced by Wordnet [Fel98], a freely and publicly available large lexical database. The main components are words, synsets, senses, and relations. Words from different parts of the speech (nouns, verbs, adjectives and adverbs) are grouped into sets of cognitive synonyms, called synsets, each representing a different concept. Given the homonyms existing in natural language, a word can be present in more than one synset, therefore, in order to unequivocally identify the synset that the word is referring to, there is the notion of word sense. Considering the "*atm*" example mentioned earlier, the word "*atm*" has two *senses*, "*atmosphere*" and "*cash machine*". Synsets are interlinked by means of conceptual-semantic and lexical relations.

We adopt the model presented in [GMFD10], where there are different instances of the linguistic component, one for each language (e.g., English and Spanish) that is to be supported by the system.

The more rigorous definition of the components of the linguistic part are:

> **Model Object 1** ($word$) A $word$ is a string of phonemes or a non-empty finite sequence of characters over a finite alphabet representing an meaningful utterance in a particular language [Fel98]. We write W to denote the set of all possible words.

Model Object 2 (Synonymous set ($synset$)) A Synonymous set or $synset = \langle SW, pt, c\rangle$ is a tuple where SW is a set of synonymous $word$s ($SW \subseteq W$) that denote the concept, represented by c, and where pt is called the preferred term of the synset ($pt \in SW$)

```
synset#1 = <
  SW={atm, atmosphere, standard atmosphere, standard preasure},
  pt=atmosphere, c#1
  >

synset#2 = <
  SW={atm, cash machine, cash dispenser, automatic teller machine},
  pt=cash machine, c#2
  >

synset#3 = <SW={machine}, pt=machine, c#3>
```

Figure 3.2: Example definitions of synsets.

Figure 3.2 shows example definitions of three synsets, where $c\#1$, $c\#2$ and $c\#3$ can be read as concept number one, two and three respectively, i.e., making the concepts language independent and where the interpretation in natural language is given by the synonymous words SW in the synset. A more formal definition of concept will be given in Model Object 6 in page 19 in the conceptual part section. As can be seen in this example, the word atm is polysemous, as it belongs to more than one $synset$, i.e., it has more than one sense. Note also that multi-word forms or expressions are also considered as atomic, when they represent an atomic sense or concept.

Model Object 3 (Natural Language Dictionary (NLD)) A natural language dictionary NLD is a tuple $NLD = \langle nlid, LW, SYN\rangle$, where $nlid$ is a unique identifier of the dictionary language; LW is a set of words of this language ($LW \in W$); and SYN is a set of synsets, such that the set of words SW of any synset in SYN is a subset of the set of words of the language, i.e., $SW \subseteq LW$.

Model Object 4 (Word sense w_s) A word sense $w\_s = \langle word, c\rangle$ is a pair, where the word is defined in Model Object 1 and c is a the sense of the word. We say that the word correspond to the Natural language part of the word sense, while the concept c correspond to the formal part, given that c is unambiguous.

For example, $\langle atm, c\#2\rangle$. Note that the word sense does not suffer from the polysemy issues, as the sense is already disambiguated.

Model Object 5 (Semantic String ($string_{sem}$)) A semantic string is defined as $string_{sem} = \langle string, \{w\_s\}\rangle$ where $string$ is the set of characters as provided by the user and $\{w\_s\}$ is a set of word senses, where each word

in the set represents a disjoint lemmatized token from $string$ representing words recognized in the linguistic part with its disambiguated concept[2].

The aim of the semantic string is to unequivocally encode the intended meaning of the string, thus eliminating possible ambiguities introduced by the natural language such as, homonyms and synonyms. This is normally achieved using Natural Language Processing (NLP) techniques, the definition of which is out of the scope of the current thesis. The interested reader is referred to [ZSG+07] for details of NLP.

3.2.2 The conceptual part

Despite the numerous research areas and researchers devoted to the definition of a concept, there is no widespread agreement of a satisfactory and complete definition of what a concept is [ML11]. The classical theory of concepts [ML11] states that *a lexical concept "c" has a definitional structure in that it is composed of simpler concepts that express necessary and sufficient conditions for failing under* c, i.e., a concept element is defined by its relation to other concepts and its relation to the world. The prototype theory extends this intuition by adding a probabilistic element to it, i.e., a concept *has a probabilistic structure in that something falls under* c *just in case it satisfies a sufficient number of properties encoded by* c*'s constituents*.

This work adopts the concept definition outlined in [APZ11a], the one defined by Description Logics (DL) [BCM+03], which we believe is a more pragmatic definition of the classical and prototype theories of concept. In DL, the semantics (or, the extension) of a concept is defined as a set of elements (or, instances). For example, the extension of the concept Person is the set of people existing in some model (e.g., in the model of the world). Because they are defined under a set-theoretic semantics, operators from the set theory can be applied on concepts, e.g., one could state that concept *Organism* subsumes (or, is more general than) the concept *Person* because the extension of the former concept is a superset for the extension of the latter concept. Among other things, the subsumption relation can be used for building taxonomies of concepts. These properties lead to a number of useful reasoning capabilities such as computing the instances of concepts through the concept subsumption, computing more specific or general concepts. A more complete account to DL is out of the scope of this chapter; interested readers are referred to [BCM+03] for details.

Pragmatically, the conceptual part is composed by concepts and semantic relations between them (e.g., is-a and part-of) as defined by [GMFD10]. Concepts are independent of natural language, where the later is considered just a mean for conveying thought [ML11], that in our model acts as a bridge between the conceptual part and humans. As humans communicate in different parts of the world

[2]For more details on tokenization, lemmatization and disambiguation the reader is referred to Section 5.2.1 in page 45 and [ZSG+07].

using different natural languages, different synsets representing the same concept in different languages (i.e., belonging to different Natural Language Dictionaries as defined in Model Object 3), are linked to the same concept.

In our example from Section 3.2.1, there are two synsets in English for $\langle\{$atm, atmosphere...$\}, c\#1\rangle$ and $\langle\{$atm, cash machine...$\}, c\#2\rangle$ (using a simplified representation). If we define the same synsets using words in Spanish, we would get something similar to $\langle\{$atm, atmósfera...$\}, c\#1\rangle$ and $\langle\{$cajero automático...$\}, c\#2\rangle$ respectively. In the conceptual part, as we can note, there are only two concepts for representing both ideas, independently of the language. Note however, that due to the well known problem of gaps in languages (i.e., given a word in one language, it is not always possible to identify an equivalent word in another language) not all concepts have a corresponding synset in all languages.

The more rigorous definition of the components of the conceptual part are:

> **Model Object 6** (Concept c) A concept c is a cognitive unit that represents the intension, i.e., the set of properties that differentiates this concept from others, and defines its extension, i.e., the set of objects that represent such property [GM10].
>
> **Model Object 7** (Conceptual relation ($concept\_relation$)) A conceptual relation is a triple $concept\_relation = \langle c_{source}, c_{target}, c_{relation}\rangle$ where c_{source} is the source of the relation, c_{target} is the target of the relation, and $c_{relation}$ is the type of relation.
>
> For example, $\langle c\#2, c\#3, c\#4\rangle$ could be understood as a relation between "cash machine" and "machine", defined by the concept c#4, which can be defined, for instance, as the is_a relation.

3.3 Schematic knowledge

Once we have the background knowledge constructs (the language and the concepts) that enable systems to interoperate, we need to defined the schema of the objects that will be dealt with in the system. The "objects" that are the subject of our framework will be called entities (see Section 3.4). In order to be able to represent these entities, we need to define the format of how these entities will look like, i.e., the types of entities that we will be able to represent in the system.

The exact definition of the entity types that are to be supported by a Knowledge Organization System (KOS) is highly domain dependent. For example, in the health domain we would need to deal with types of entities such as Persons that can act as Patients or Doctors, Diseases, Treatments and Medicines, which are not the same of types of entities that are dealt with by the music domain. However, we need to define the basic building blocks that will allows us to define entity types uniformly across different system.

The basic elements of the schematic knowledge are entity types and attribute definitions. The definition of these components are:

Model Object 8 (Entity Type ($etype$)) An Entity Type (etype) is a tuple $ET = \langle c, \{AD\}\rangle$, where c is the concept of the etype, and $AD = \{AD\}$ is a non-empty set of attribute definitions.

We envision specifying a set of mandatory attribute definitions that have to be present with their corresponding values when an entity of the particular $etype$ is instantiated, . This however, is highly etype dependent, and furthermore, domain dependent, e.g., while in a medical application the gender of a person might be mandatory, in a phone application most likely it is not strictly mandatory. In general, the $etype$ definition is the entry point for the creation of an entity, and the set of attribute definitions should normally be shown to the user creating the entity[3].

Model Object 9 (Attribute definition (AD)) An Attribute Definition is a tuple $AD = \langle c, D, isSet\rangle$, where c is the concept that represents the attribute name, D is the domain of the data type that restricts the possible values for the attribute definition (i.e., date, string, integer, real, relations, concept ...), and $isSet$ is a boolean value which indicates whether AD can be instantiated to a set of values (in the case $isSet = true$) or to a single value (in the case $isSet = false$). We choose to store the concept in the definition and not the natural language name in order to avoid the ambiguity of the natural language is its related problems [APZ11a].

Considering D, we can distinguish betweem three types of domain data types:

normal (semantic-less) data types such as string, date, integer, float ... . These are normal basic types that are not linked to a concept in the conceptual part.

- conceptual values including semantic strings and word senses, i.e., those which have a formal part with a link to a concept in the conceptual part. Selecting a specific concept as the domain allows us to restrict the possible values even further, for example, when defining a color attribute, we can set the domain to the concept of color. By doing so, all and only those concepts which are colors (defined by the is_a relation between the concept of color and the colors such as black, white, red...) will be allowed as values of the specified AD.
- relational attribute values, which means that the value of the attribute is a reference to an entity, for which the entity type is defined in when specifying the relation, e.g., when defining a friend attribute, we can say that the type of the entity has to be a Person.

As stated before, the exact entity types to be modeled are domain dependent. However, there are some common entity types that are usually shared across sev-

[3]How to show the set of attributes to be filled in by the user is an Human Computer Interaction (HCI) issue, which is out of the scope of this thesis.

eral application domains. These entity types could be defined a-priory fostering reusability, thus avoiding heterogeneous definitions for the same type. Some common entity types are, for example, Person, Location, Organization, Event and those more specific than Mind products such as Music, Paintings, Pictures, etc. We develop a root entity type called "Thing" which can be used as default entity type for any entity, in the case the domain application wants to leave the entities untyped (e.g., OKKAM [BSG07])

A recent joint effort by several Internet search engines have made available at schema.org[4] the definition of these entity types, and other commonly used types of content for the web, allowing webmasters to markup their pages in order to structure the data, that can later be reused by others and be recognized by these search engines in order to improve their search results.

3.4 Concrete knowledge

The purpose of our framework is to manage the knowledge of the users and communities. While the language and concepts are important in order to share knowledge, they are not the aim, per-se, of most common Knowledge Organization Systems; they are just means by which we are able to exchange of knowledge and ideas. When people engage in conversations, they talk about things, and these things, are what we will denominate *entities* in our framework. The types of entities, or their schema, is defined a priori, and the general model for defining these entity types is presented in Section 3.3.

An *entity* is any object that is important enough for a person to talk about it and give it a name (e.g., a Location, a Person, an Event...). We make the distinction between *real world entities* (those that exist), mental entities (those that people form in their minds about real world entities) and digital entities (those that we model in the system trying to capture the semantics of the previous two).

Given that diversity is an unavoidable and intrinsic property of the world and as such cannot be avoided [GMD12], the ultimate purpose of the semantic framework is to interoperate at the level of entities, considering people's own diverse views of the world, by being able to manage the identities of the *real world entities*.

3.4.1 Real world entities

A (real world) entity is something that has a distinct, separate existence, although it need not be a material (physical) existence[5] that belongs to a certain type [YYSL08]. Each real world entity has a set of properties that evolve over time in the course if its existence, as it interacts with other entities and the world. For example, people (Person entities) might move from locations to locations, their physical appearance

[4] `http://schema.org/`

[5] `http://en.wikipedia.org/wiki/Entity`

change over time (see Figure 3.1), they might acquire or loose titles (such as engineer or baron) and play different roles at different times such as actor or governor. This evolution though, might not always be perceived by everyone aware of their existence at all times.

$$(RW)E = \langle \{A^T\} \rangle \tag{3.1}$$

Equation 3.1 shows a model of a real world entity $(RW)E$ containing a set of attributes $\{A^T\}$ where the $T = [t1, t2]$ stands for the time interval when the values of the attributes are valid, i.e., as the entity evolves, the same attribute might have different values, for example the $hair\_color$ attribute might change from "red" to "dark brown" as can be seen in Figure 3.1, and new attributes might appear and disappear.

From the point of view of the existing real world entity, its uniqueness is given by its existence and expressed to the world by its attributes and its actions.

3.4.2 Mental entities

A mental entity is a personal (local) model created and maintained by a person[6] that *references* and *describes* a real world entity [7]. According to Searle [Sea58], the *reference* is achieved by a proper name and the *description* is achieved by definite descriptions. A definite description [Rus05] is a characterization that entails the existence, uniqueness and universality of the entity (or object) being referred [Lud11], i.e., it is a set of uniquely referring descriptive statements.

$$(M)E = \langle name, def\_desc \rangle \tag{3.2}$$

Equation 3.2 shows the tuple conforming the mental entity $(M)E$ that is composed by a $name$ (to be further discussed in Section 3.4.2.1) and a definite description def_desc (to be further discussed in Section 3.4.2.2).

3.4.2.1 Mental identifiers: names

Names are (local) symbols created by people to refer to specific (real world) entities. Each (real world) entity being identified by a name has a (local) set of characteristics that distinguishes them (locally) from the other real world entities of the same type (called a definite description in Philosophy). The pragmatics of a proper name lies in the fact that it enables us to refer publicly to real world entities without being forced to an *a-priori* agreement on the descriptives characteristics exactly identifying the real world entity being referred [Sea58]. Thus, what differentiates

[6] This applies to agents in general such as people and organizations, but for the sake of brevity and clarity, we will us the term person instead of agent throughout the document.

[7] Our use of mental entities here should not be confused with fictional entities as used in philosophy, i.e., entities that are not real, e.g., Batman. What we mean is a personal model of real world entities, whether these exists or not.

(real world) entities from one another[8] is encoded in the definite description, and the name is a reference to this description [Sea58], *an identifier* created by people to avoid having to exchange the whole definite description when interacting with each other.

Names used in mental entities have the following characteristics:

- A name is a (local) identifier for a definite description [Sea58];
- Both the name and the definite description in one mental entity have the same referent (the same real world entity) [Sea58];
- Names are locally created and maintained by people, which entails that:
  - there can be different names (created by different people) for the same real world entity (synonames, simmilarly to synonyms).
  - the same name can be given to different real world entities (homonymy).
- There can be different (local) names, for the same (local) definite description.

Note that the $name$ element is by definition a reference to the real world entity being mentioned or thought about. Most commonly, proper names play the role of the $name$ element, and whenever this is true, we can directly recognize the referent. However, we also normally use "indexicals" to refer to real world entities, i.e., context-sensitive expressions whose meaning vary from context to context [Bra10]. Example indexicals are "my mother" or "my home town", each having a different referent according to who is uttering the expression. Nicknames are other types of names that are typically used. There are also cases when arbitrary "labels" are used as names. A person might use one or more of these kinds of names to refer to the same real world entity. Whenever we use the element $name$ in this document, we refer to all these classes of names, and possibly others, that serve as references for *real world entities* in the mind of the people.

3.4.2.2 Descriptions

A (definite) description is a (local) set of characteristics, a mental model created by a person, that (locally) distinguishes real world entities from one another.

Multiple mental entities (created by multiple people) can refer to the same *real world entity* from different perspectives (according how the person has interacted with the *real world entity*) encoding different (overlapping or disjoint) descriptions. For example, multiple people might have different points of view of a particular person (there is only one person in the real world being referred), perhaps because they know the person from different social contexts, e.g., Arnold Swarchenegger as an actor, as a politician, as a friend, or as a father.

Mental entities and their corresponding definite descriptions have the following characteristics:

[8] from the point of view of a person

1. A mental entities is a local model of a real world entity (it encodes one point of view);

2. Different mental entities from different people with different perspectives, can refer to the same real world entity;

3. Mental entities are *incomplete*, as the person creating the mental model does not know all the details about the real world entity[9];

4. Mental entities can have *inconsistent*[10] descriptions, mainly due to:

   (a) temporal dependency, i.e., a description that is not true anymore, but which was true in some past, e.g., "*the person with black hair*". This is due to the evolution of the entity and the fact that descriptions act as static "photos" of real world entities at one point in time [11];

   (b) subjectivity, e.g., nice person, horrible place;

   (c) false descriptions, those that were never true about the real world entity, but is believed to be true by the person holding the description, e.g., Schwarzenegger was the president of USA (only true in a fictional movie)[12].

5. The uniqueness of reference of the definite description in a mental model is considered with respect to all the other entities the bearer of the mental model has, not with respect to all the other possible real world entities of the same type [Sea58];

6. The definite description of a mental entity might be formed via a direct interaction with the real world entity it refers to, or learned from someone else.

7. The mental description is created using the local knowledge[13], therefore different descriptions for the same characteristic might be different. E.g., one might think that someone's eye is blue, but another person might say that is marine blue (just because s/he has a more detailed knowledge about colors);

3.4.3 Digital entities

The purpose of this work is to develop a model that captures the semantics of the real world entities. Given that real world entities can only be captured and modeled

[9] Unless the mental entity is about him/herself, but not all entities are self-conscious.

[10] with respect to the real world entity

[11] This is why the time validity T is important in equation 3.1.

[12] Note that given the local knowledge and context of the person with this description, this might be true, but nevertheless, it is universally false, i.e., not true about the real world entity.

[13] local language and concepts which might differ from person to person

by people[14], we also need to encode and understand the local semantics of the mental entities that are provided by people, which can differ from one another, being referred and described in diverse ways and even in different languages [GMD12]. Digital entities are the way we capture real semantics by allowing the computer to bridge between the mental entities people provide about the real entities[15]. In the following models we will make clear how we capture each of the important elements of the different kinds of entities. In the rest of this document we will use the term entity to refer to *digital entity*, and whenever we refer to one of the other two (real world or mental entities) we will make an explicit distinction.

In order to capture the real world semantics of real world entities, as provided by the mental entities, we need to address the following issues present in mental entities:

1. The name as identifier is locally unique, but not globally unique (homonyms, indexicals).

2. The same real world entity can be given different names (synonames).

3. A mental entity encodes only a partial view of the real world entity (this is known as the *partial knowledge problem*).

4. Neither names nor definite descriptions are persistent in the real world entity (they evolve over time as the real world entity evolves).

5. Definite descriptions are only locally identifying (at the mental level). This can create two types of problems when people refer to real world entities:

    - People believe they talk about the same real world entity, while they are actually referring to different real world entities (false positive FP);
    - People believe they talk about different real world entities, while they actually refer to the same real world entity (false negative FN).

The above mentioned issues uncovers the need of being able to unequivocally encode the identifier of real world entities, given the diverse views over them, also being able to encode and maintain these different views. This means that we need two identifiers for the digital entities: *i)* a real world identifier, which captures the real world semantics of the real world entity, referred to as SURI (from semantic URI), and *ii)* a mental identifier, which allows the identification and subsequent retrieval of a particular point of view, referred to as SURL (from semantic URL).

Figure 3.3 shows the relation between the elements of equations 3.1, 3.2 and how these elements are mapped into the main components of a digital entity. As we can see in Figure 3.3, a digital entity (E) contains a SURI[16], which represents

[14] Sensors can also give automatic information about entities, but still they will be programed by people and thus use the mental model of the person programming the sensors.

[15] A digital entity can be thought as if it was the mental entity of the computer.

[16] A more complete discussion about SURI is presented in Section 9.2.1.2

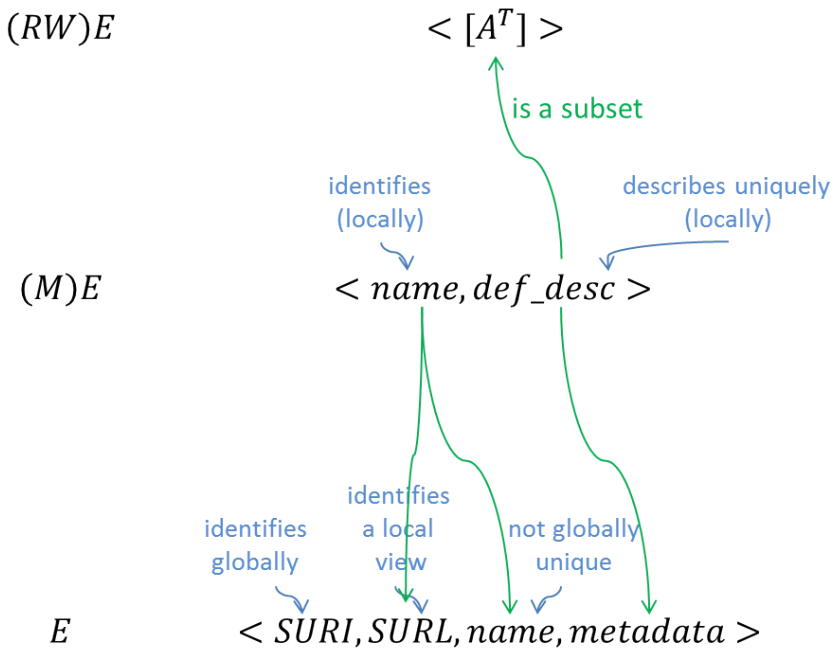


Figure 3.3: The relation between the elements of real world, mental, and digital entities.

the identifier at the real world level, and a SURL[17], which represents an identifier at a local (mental) level. Given different local views, defined by the metadata and identified by the SURL, the system should be able to distinguish that these local views (with different SURLs) actually refer to the same real world entity (as identified by the common SURI).

In what follows we present the model used for representing the digital entities, which incorporates the desired characteristics presented in the previous sections. We use a recursive style for defining the digital entity, where each new element in a Model Object will be defined in a subsequent Model Object. Model Object 10 introduces the model for the digital entity, that is to be used as the basis of our framework.

> **Model Object 10** (Entity) An entity e is a tuple $e = \langle$ *ID, SURL, etype, name,* EM, $[link], [copy]\rangle$, where:
>
> > ID is a universally unique identifier of the *real world entity* modeled by e *(which globally encodes the uniqueness and existence of real world entity)*. Only when the global uniqueness condition over EM is met (see Section 8.2.2) the $ID = SURI$ (in Chapter 9 we will expand on the definition of ID with Model Object 14);

[17] A more complete discussion about SURI is presented in Section 9.2.1.1

- $SURL$ is the local (dereferenceable) URL identifying the entity *(which corresponds to local identifier in the mental entity without the homonimy problem*, see Section 9.2.1.1);
- $etype$ is the entity type (e.g., Person, Organization, Event);
- EM is the Entity Metadata encoding the local mental model of e *(which corresponds to the (local) definite description created by a person using his/her knowledge, i.e., language and concepts, and is it a subset of the evolving attributes $< \{A^T\} >$ of the (RW)E in equation 3.1)* taken at a certain point in time;
- $link$ is an optional property that refers to a remote entity if the person creating the entity decides to share the metadata definition EM of another person, i.e., to fully rely on the remote entity being referred for its metadata definition EM, *(which corresponds to the case when people decide to fully trust other's mental entity definition)*;
- and $copy$ is also an optional property that specifies from where this entity has been copied. *(which encodes the way we interact with the real world entity, remembering if we have had learned about this entity from someone else or not)*.

The last two elements of the model, $link$ and $copy$ allows us to solve the real world phenomena of knowledge propagation between people in two different ways: *i)* in real time by sharing (see Section 9.3.3), and *ii)* by learning from other sources of information by copying (see Section 9.3.2).

By being able to *share* entities we can significantly reduce the number of local entities with different metadata (EM) and IDs to maintain, align and match. Well known entities could be maintained by trusted people, organizations or communities and other people can reuse these trusted definitions. For example, entities of type Location (e.g., countries and cities) and famous people (e.g., politicians, actors, scientists) could by maintained by dedicated communities. By allowing to *copy* entities, and therefore fully import the whole entity definition encoded in EM, we allow people to quickly learn from one another, and consequently improve upon EM.

Model Object 11 (Entity Metadata (EM)) The Entity Metadata (EM) is a set of instantiated attributes $EM =< \{Attr\} >$ where each $Attr$ defines a characteristic of the entity e with its value(s). The $type$ definition in the entity contains the list of attribute definitions AD that should be suggested to the user, some of which might be mandatory depending on the application domain.

Model Object 12 (Instantiated Attribute ($Attr$)) An instantiated Attribute is a tuple $attr = \langle AD, AV, T \rangle$, where AD is the attribute definition (see Model Object 9), the attribute value AV is an element or set of elements

(according to $isSet$ from AD) from the domain D of possible values as defined in AD, and T is a time interval during which the assignment of AV to AD is valid, if applicable. The possible AD to be used in a particular entity is recommended by the set of attribute definitions $\{AD\}$ available in the $etype$ (see Model Object 8) of the entity for which the current instantiated attribute $Attr$ is being created.

3.5 Related Work[18]

In this chapter we presented a model for managing knowledge that addresses several sources of semantic heterogeneity by construction. Therefore, instead of focusing on post-hoc solutions to the semantic heterogeneity issues and therefore interoperability, we focus more on how to deal with heterogeneity by design in the knowledge model. The post-hoc approach for dealing with heterogeneity is spread among several well established research areas such as schema matching, ontology alignment, database interoperability, all of which, are out of the scope of the current chapter. In this section we will analyze other models of Knowledge Organization System (KOS) and how these compare to our solution.

There are several KOS models currently in use by people and organizations. These KOSs can take the form of an authority file, a glossary, a classification scheme, a taxonomy, a thesaurus, a lexical database (such as WordNet), or an ontology, among others (see [Hod00] for an in-depth discussion on the different types of KOSs). Figure 3.4 shows different kinds of ontologies with different levels of expressivity and formality which can vary from a list of terms, through Database schemas to rigorously formalized logical theories using Description Logics [UG04]. In what follows we will compare and contrast some of these Knowledge Representation Systems to the one presented in this chapter.

Relational Databases are one of the most commonly used KOS. They are based on relational algebra and use tables, columns, primary and foreign keys and constraints to implement the knowledge model of the domain. There is a close relation between the schematic knowledge presented in Section 3.3 and the database schema (tables and columns). The concrete knowledge (see Section 3.4) is comparable to the rows of the tables (the data itself), which are the instances of the given schema. Despite these similarities, current relational database systems have no sufficient support for the Background Knowledge model presented in Section 3.2. There is no notion of language independent concepts that can be used to model the schema, nor to be used as values. The semantics of the schema usually does not become part of the database definition and is lost after design-time. This is one of the mayor reasons contributing to the heterogeneity problem [PH00], which hiders the interoperability between different database systems, even when dealing with

[18] Parts of this section were published in [AZP12] in collaboration with Pierre Andrews and Ilya Zaihrayeu.

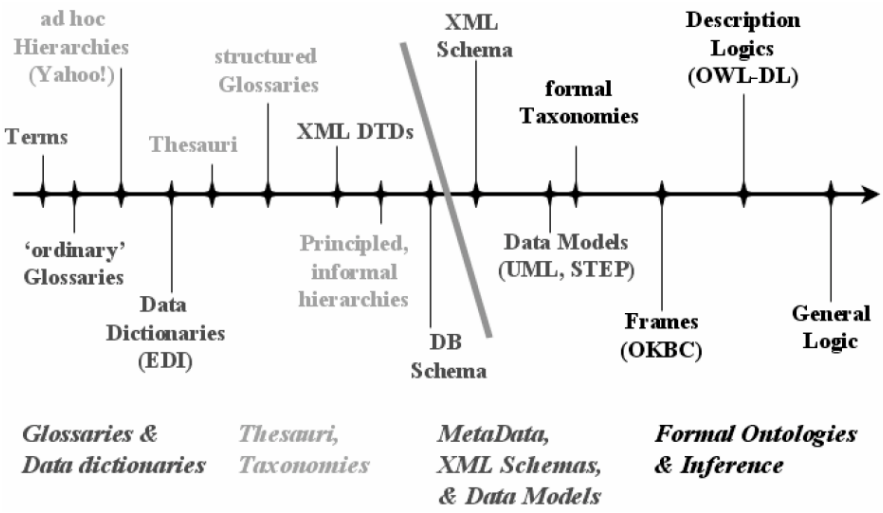


Figure 3.4: Kinds of ontologies from [UG04].

the same domain. This is the main problem that the Schema Integration field has been dealing with for many years [BLN86].

The knowledge model proposed in previous sections has a direct formalization in Description Logics (DL) [BCM$^+$03]. DL classes correspond to concepts, instances to entities, relations and attributes to roles [GDMF11]. The Conceptual part (Section 3.2.2) and the Schematic Knowledge (Section 3.3) provide the TBox, while the Concrete knowledge (Section 3.4), i.e., entities, provides the ABox for the application domain. However the linguistic part (Section 3.2.1) cannot be properly represented in DL. DL identifiers are only symbols used to denote an element of the domain of interpretation. As such, it does not allow the definition of multiple synsets in different languages for a particular concept, although a name predicate could be attached to each identifier, this has not the same power of expressiveness as multilingual synsets.

An Ontology, as defined by Gruber et al., "is an explicit specification of a (shared) conceptualization" of the application domain [Gru93]. In practice, ontologies are usually modeled with (a subset of) the following elements: concepts (e.g., CAR, PERSON), relations between concepts (e.g., PERSON is-a BEING), instances of these concepts (e.g., bmw-2333 is-instance-of CAR, Mary is-instance-of Person), properties of concepts and instances (e.g., PERSON has-father), restrictions on these properties (e.g., MAX(PERSON has-father) = 120), relations between instances (e.g., Marry has-father John), etc [Gro04]. These elements have a direct relation with the Model Objects presented in Sections 3.2, 3.3 and 3.4. However, depending on the specific implementation, the duality between the linguistic and conceptual part presented in Section 3.2 is captured or not.

Some languages for authoring ontologies are currently defined by the Semantic Web community, implementing the vision by Tim Berner-Lee [BLHL01], sup-

ported by WorldWide Web Consortium (W3C)[19] with participation from a large number of researchers and industrial partners. Examples are the Resource Description Framework (RDF) [LS99] and the OWL Web Ontology Language family of languages [MvH04].

While the model presented in the previous sections has a direct relation with the elements of OWL, and thus can be implemented with it, there are some issues when comparing it to RDF. In RDF there is no explicit distinction between concepts and individuals, therefore we cannot neatly distinguish between classes and their instances (concepts (see Model Object 6) and entities (see Model Object 10) as defined in our model). Furthermore, in RDF there are no cardinality constraints when assigning predicates to instances. Considering our previous example, nothing prevents us from creating two predicates such as "Mary has-father John" and "Mary has-father James" which cannot be the case considering real world semantics, i.e., assuming that the semantics of *has-father* is biological father, that Marry refers to the same real world person, and that James and John are different real world people. The previous example also shows another drawback of defining the predicates (attribute definition in our terms) with natural language labels, i.e., it is not uncommon to find subtle differences in meaning between any two ontologies even if they represent the same domain and are encoded in the same formalism [SK08]. For example, the *price* element in one ontology could include taxes, and in another could mean without taxes. Finally, given that the assignment of URIs to instances (entities) is optional within RDF, metadata for entities can be created without a link to a real world entity [HHD07].

3.6 Summary

In this chapter we propose a Knowledge Organization System (KOS) centered on the management of entities, that by construction deals with several sources of semantic heterogeneity. The background knowledge deals with the issues of the ambiguity of natural language converting words into formal concepts. Based on the background knowledge, the schematic knowledge defines the model for entity types that can be instantiated and reused according to the application domain.

The main contribution of this chapter is the modeling of the digital entities, in the concrete knowledge, that captures the real world semantics of the real world entities (those that actually exist) as provided partially by people from different perspectives using diverse terms and languages. The digital entity solves the problems with names and descriptions of (mental) entities provided by people by relying on the SURI (semantified URI) to identify the real world entity while still allowing the identification and dereferencing of the different perspectives by relying on SURLs (semantified URLs). The process that specifies how to assign the values of these two identifiers is the core of Chapter 9: "Identity Management".

[19] `http://www.w3.org/2001/sw/`

In the following Chapter we will see how these components are used to achieve interoperability at different levels. The background knowledge encodes well studied language elements in a top-down approach, from the universal level to communities, and from communities to people; while we foresee that entities are going to be created mostly in a bottom-up approach, from people to communities.

An important issue with the background knowledge is the coverage of the language, i.e., how many words we are able to recognize. An initial approach can be that of bootstrapping the set of words of the linguistic part with Wordnet (the original English version). However, given the dynamic nature of knowledge, we cannot guarantee full coverage using a static one-time importing approach. In order to deal with this issue, in Chapter 6: "Background Knowledge Evolution", we define the process by which the background knowledge is updated following a bottom-up approach too.

Chapter 4

The semantic layers

4.1 Introduction

Semantics is a local and a global problem at the same time. Local because is in the mind of the people who have personal interpretations, and global because we need to reach a common understanding by sharing and aligning these personal interpretations.

Current approaches of knowledge management, ontology building, ontology maturing and semantic interoperability (matching, alignment) seem to focus on the existence of two knowledge layers, one local, where the person creates and shares knowledge, and a global (sometimes also referred to as universal) which is normally designed to foster interoperability. We argue that while this is a possible scenario, there is a no clear understanding of what being global means, and with respect to what. Several research use the terms "universal" and "global" to actually refer to the (social) context in which they are being applied, i.e., the whole organization, the whole community, and not actually the whole world. The terms are indeed, rarely used to refer to "everyone", including those who are outside of the context of the organization or community.

Understanding the above mentioned difference is very important, as it is the way we believe knowledge evolves. Rather than in a two-layer fashion, we argue that knowledge evolves in a three-layer architecture, where in the first (bottom) layer people create and maintain locally their own vision of the world (or the part they are interested in), then they share this knowledge with other people and interact in a bounded (social) context (an organization, a community, a group)[1], but in order to achieve successful multi-context interaction, there is the need of a higher layer that supports interoperability and that formalizes what is commonly known across social contexts (communities), a Universal layer. For example, there is no disagreement of what a *person*, *gravity*, *Newton* (the scientist) or a *cat* is.

Although some approaches start to go in this direction creating rather minimal

[1]Notice that a person can belong to multiple communities, organizations and groups at the same time

upper ontologies such as DOLCE [MBG+02] or more rich upper and domain ontologies such as SUMO [PNL02] and SWIntO [OAH+07], these approaches seem to believe there is a precise finite knowledge state that can be captured and formalized. This "simplified" close world assumption works in research scenarios and closed scenarios, but the reality is that knowledge is in constant evolution [SK05] and there is a real need to add new concepts, terms and instances recognizing the place of people in the process of creation of knowledge (a bottom-up approach).

In the following sections we will see that the Components of the knowledge presented in Chapter 3 fit into each of the three layers, and explain the role they play at each layer. In Section 4.2 we present our approach to the universal layer, called Universal Knowledge (UK), that aims at encoding the language and concepts needed to achieve interoperability at different layers, separating the knowledge in domains; in Section 4.3 we see how communities can reuse the parts knowledge that suit their purposes, being able to define new knowledge that evolves faster in their context and that is reused by all the members of the community; and in Section 4.4 we present how at the personal layer we allow users to reuse the knowledge by bootstrapping from the UK and becoming members of communities, allowing users to define their own local terms, concepts and entities. Section 4.5 summarizes the Chapter.

4.2 Universal (global) layer

The main purpose of the Universal Knowledge (UK) layer is to maximize the compatibility between users, allowing them to easily interoperate while exchanging knowledge. This is done by trying to minimize the semantic heterogeneity between them in a top-down fashion. As observed in [PH00] one of the main components of the heterogeneity comes from the ambiguity of natural language. As we saw in the Chapter 3 "Knowledge Components", the Background Knowledge component deals with this ambiguity by translating natural language into a formal language, the conceptual part. These two components are the most important components at the UK as they give the basic constructs that can be reused by different communities to further define the knowledge in their domains.

The content of the Background Knowledge of the UK can be initially bootstrapped from widely used thesaurus and dictionaries which are Language dependent such as Wordnet [Fel98] or Moby Thesaurus[2]. The Linguistic part of the background knowledge is usually regulated by Linguistic Institutions (each language has an institution dedicated to study the new terms to be allowed in their dictionary), therefore the decisions made by these institutions governs the terms that are universally accepted for each language.

Another important source of heterogeneity are the schemas used to model the entities. At the UK layer we encode a basic set of entity types such as Person, Organization, Localtion, Event. These can be reused and further define by each

[2] `http://icon.shef.ac.uk/Moby/`

user and community to suit their needs. Defining a basic set of entity types based on the linguistic and conceptual part of the background knowledge allows us to address much of the schema heterogeneity issues in a top-down fashion, while alleviating the work of having to define new schemas from scratch when needed.

Given the vast amounts of concepts and linguistic elements in the background knowledge of the UK, we follow the approach presented in [GD11] to split the knowledge into domains in a hierarchical fashion. The content of each the domain is considered to be well studied by experts in each field and widely accepted between all the users. One initial source for the Domain Knowledge can be the knowledge encoded in library science. For example, consider the extract of a classification for medicine taken from the Library of Congress Classification[3] in Figure 4.1. the domain of Medicine could be composed of several sub-domains, in the example, Surgery, Ophthalmology, Nursing and others. Each sub-domain can be composed of other sub-domains, as for example Surgery contains Plastic Surgery, Surgical Nursing and others.

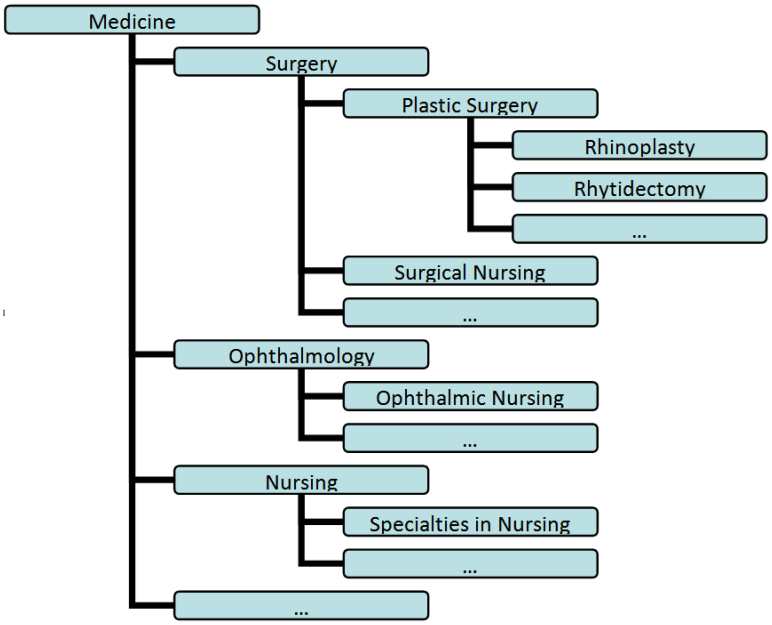


Figure 4.1: Extract from the Library of Congress Classification for Medicine.

Given this domain structure of the knowledge, users and communities can choose which part of the knowledge encoded at the Universal layer to import into their local instances of the knowledge base.

Similar approaches to the notion of Universal knowledge have been defined, to some extent, by DOLCE [MBG+02] and SUMO [PNL02]. DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) is a minimalistic foundational ontology that serves as basis for developing other ontologies. While being rigorous,

[3] http://www.loc.gov/catdir/cpso/lcco/

the minimality principle followed by the DOLCE makes it only a starting point for developing other ontologies, leaving again the work of defining the terms, the relation between terms and the structuring of the schemas of the application domain to each application developer. This freedom allows different views of the domain, opening the door to heterogeneous ontologies, even if based on DOLCE.

SUMO (Suggested Upper Merged Ontology) was created merging a number of existing upper level ontologies from different domains. SUMO concepts contain links to Wordnet synsets in a attempt to improve language coverage. SUMO is the most similar work to our understanding of the universal layer, however, we aim at a much higher language coverage (we bootstrap the linguistic and conceptual part of the BK with all the linguistic information from Wordnet, not just part of it) and make a clear cut between the concepts which are completely language independent and the words that can be used to describe concepts in each of the supported languages, while in SUMO the concepts are defined in English.

Furthermore, neither DOLCE nor SUMO, define the approach for managing the identity of the entities (instances of the concepts) being defined with their ontologies [BSG07]. They approach the issues at the background and schematic components of the knowledge, while completely ignoring the concrete knowledge as defined in Chapter 3 "Knowledge Components". Finally, neither of them tackles the issue of the dynamic nature of knowledge, while we define in Chapter 6: "Background Knowledge Evolution" how to update the background knowledge in a bottom-up manner.

4.3 Community (glocal) layer

A community "*is a group of people who share interest for a given topic, and who collaborate to deepen their knowledge through ongoing learning and sharing*"[4]. Considering real world communities as metaphor, we assume that individuals participating in communities gain knowledge which is specific to the topics the community is formed around. This knowledge includes vocabulary terms (e.g., the term "rhinoplasty" to refer to "cosmetic nose surgery" in medicine), entities (e.g., the members of the community), etc.

The goal of the community layer is to sit between the Universal layer and the Local (personal) layer. It is local from the point of view of the Universal layer since it can localize the knowledge around a certain topic, allowing the participants of the community to manage this knowledge without requiring universal agreement. This locality is important as it gives a bounded context for the creation and exchange of new ideas. These local (to the community) ideas can be adopted by all the members of the community, and later be shared across communities, to become Universal. It acts as global from the point of view of the Personal layer because

[4]The definition is from knowledge communities defined by aia.org (`http://www.aiasww.org/resources/knowledge.htm`)

it allows the participants of the community (and perhaps similar communities) to share knowledge and communicate ideas with each other.

The importance of this layer can be appreciated with the following example: nowadays when referring to Java one normally thinks about the "programing language"; however, when this term was first used to refer to this relatively new meaning, the term was already widely known to refer to the "coffee" or the "island" meanings. Only in the context of the community around the new programming language being developed (Sun Microsystem) the new meaning was known. Later, this new meaning was spread to other communities of universities and taught as a programming language, used in many applications and became well known, to the point that now is widely known and can be included at the Universal layer.

Considering the knowledge components presented in Chapter 3, each community has its own background, schematic and concrete knowledge. The interoperability can be achieved by reusing the knowledge defined at the Universal layer. Each community can bootstrap its knowledge importing the domains that are deemed useful from the Universal layer, and can further extend each of the three components with new terms, concepts, entity types and entities. This allows flexibility and decentralization in the creation of knowledge, while still trying to reduce semantic heterogeneity. For example, a community around surgery can import the part from the Surgery domain from Figure 4.1. For each of the medical terms, it can extend the linguistic part of the background knowledge to include the commonly used terms such as "nose job" for "rhinoplasty", and further extend the schematic knowledge for Person (that can be taken from the UK) to the entity type of patient, including and maintaining entities for the known surgical clinics and surgeons.

The difference between the community knowledge and the domains encoded at the Universal knowledge is that a domain is a subset of the Universal knowledge that can be reused across several communities. One community, however, can reuse different domains to structure the knowledge around the topic of interest. In the example of the paragraph before, it is clear the besides the domain of Surgery the domain of Space (that include the terms, concepts, schemas and entities related to locations) is also needed to specify the location of the entities for the surgical clinics and the surgeons. The power of this difference resides in the fact that several domains can be reused at different communities, thus fostering interoperability, while allowing the community to extend the knowledge locally to foster interoperability among the members of the community, allowing flexibility for new knowledge to emerge.

Note that we did not define the mechanism by which new knowledge is created at the community layer, since this will depend on the type of communities and the norms governing them. For example, a "democratic" community could use a voting mechanism to reach agreement between the participants. In another type of community there can be an administrator that is in control of all the knowledge of the community. In the later case, users already "pre-agree" with the knowledge of the community, since they agree to join the community and to follow its rules at

subscription time.

4.4 Personal (local) layer

The goal of this layer is to allow people to manage their knowledge locally, thus avoiding privacy issues that might arise when user's data is stored in a central server or service not owned by them. People would normally want to manage, organize and manipulate *things* they are interested in according to their own personal point of views, based on their perception, experience an interactions with other people in diverse social contexts. People and may belong to zero, one or more communities as can be seen in Figure 4.2.

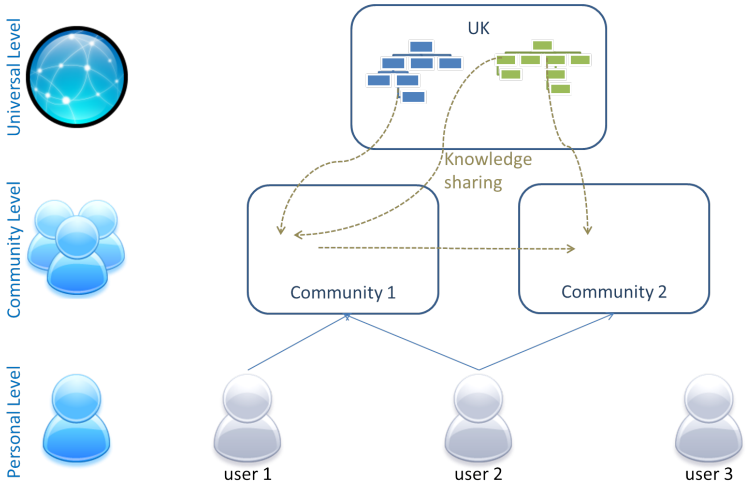


Figure 4.2: The connection between the three layers: Universal, Community and Personal

As we saw in the previous Chapter, these *things* people care about are modeled by entities in our knowledge base framework (see Section 3.4). In this distributed scenario, the workload of the maintenance of the entities that are of interest (at the personal later but also at the community layer) is distributed among all the users of the system, requiring no explicit a-priory agreement. However, in order to fully exploit the distribution of the work, and the fact that others can benefit from already created entities, we need to provide interoperability between entities.

Given the locality of entities, we expect that they will be mostly created in a bottom-up fashion, therefore we keep the entities were they are generated and used the most (at the local layer). However, in order to achieve interoperability and enable the power of the distributed system, we provide the Universal and Community layers, from where people can take shared conceptualizations and reuse them as much as possible. This means that while entities are created in a bottom-up fashion, the language, concepts and the schemas used to define and describe the

local entities are given more in a top-down fashion, coming from the Universal and Community layers, as it makes little sense to ask people to create from scratch the whole background and schematic knowledge presented in Chapter 3 “Knowledge Components”.

While we achieve interoperability by reusing background and schematic knowledge from the Universal and Community layers, we also need to support a process by which these types of knowledge are formed in a bottom-up fashion. This is because knowledge is in constant evolution, and new terms, concepts and types of things that we care about are normally created by people, as we saw in the “Java” example of the previous section. Given this bottom-up emergence of knowledge, we need operations to evolve the knowledge shared by a group of people, based on the usage in this distributed environment. Furthermore, once a new piece of knowledge is created manually or automatically, we need to propagate this knowledge between users, between users and communities, and between the communities. The process of automatic evolution of knowledge is studied in Chapter 6: “Background Knowledge Evolution” and the process of propagation of new knowledge is presented in Chapter 6: “Background Knowledge Evolution”.

Given the diverse perspectives of entities provided by people, there is a need to manage the identity of what these entities (in the system) are referring to (outside the system). This means that once we are able to interoperate at the level of language and schemas, we need to be able to interoperate at the level of entities, i.e., to know when we talk about the same thing. For this we distinguish between the real world entities (the things we refer to), the mental entities (the personal view about the thing we refer to) and the digital entity (the way we encode our vision in the system). Given the digital entity, we need to be able to compare them and to know when they refer to the same real world entity. The compatibility conditions are presented in Chapter 8: “Entity Compatibility” and the management of the identities of the entities is defined in Chapter 9: “Identity Management”. In the later Chapter we center our attention in capturing the semantics of the real world entities as provided by people, therefore dealing by construction with different points of view. We therefore recognize the difference between the identification of the real world entity and the identification of the, possible different, (partial) views provided by people.

4.5 Summary

In this charter we presented the different layers that we use to deal with the semantics problem: Universal, Community and Local. The key difference from other state of the art work is the introduction of the Community layer, which represents a bounded context where new knowledge can evolve according to the needs and interest of the participants of the community. The community layer can reuse (parts of) the knowledge defined at the Universal layer, therefore enabling interoperability between communities. At the personal layer users can create and maintain their

own local knowledge according to their own perspective in a distributed manner. This avoids the privacy issues that are present when knowledge is only maintained in a central repository that is not owned and managed by them.

In this distributed scenario, we expect entities to be created and maintained mostly by people (bottom-up) while reusing the linguistic and conceptual parts of the background knowledge and the schematic knowledge from upper layers, i.e., their communities and the Universal knowledge. This means that normally the background and schematic knowledge are created and propagated in a top-down fashion. However, we saw that, given the dynamic nature of knowledge, there is still a need for allowing linguistic, conceptual and schematic knowledge to be created also in a bottom-up manner and to be propagated between users, between users and communities and between communities. These operations will be studied in Chapter 6: "Background Knowledge Evolution" and in Chapter 7: "Knowledge propagation".

The problem of reaching interoperability at the different layers considering the language, concepts and schemas is the core of Chapter 5: "Semantic Matching"; while the problem of reaching interoperability between entities is the core of Chapter 8: "Entity Compatibility" and Chapter 9: "Identity Management".

Part III

Background and Schema compatibility

Chapter 5

Semantic Matching

5.1 Introduction

Matching is a common technique used to address the semantic heterogeneity problem arising when systems with diverse data structures and languages expressing different viewpoints wish to interoperate. Some application domains that benefit from this technique are semantic web, data exchange, query translation, data integration [Hal05], peer to peer networks, agent communication [ES07] and many more. The matching process takes as input two ontologies or schema-like structures and produces as output a set of alignments between the corresponding elements [GYS07].

Depending on the domain, the input structures can be among classifications structures (business catalogs, web directories, user directories in the file system), database or XML schemas and ontologies. The information in these structures is normally described using natural language labels. Figure 5.1 shows a subset of Google and Yahoo Web directories related to movies. If a movie oriented application wants to show an integrated result, e.g., the list of actors, it is clear that it would need to match these two directories, and use the information they contain. In the example of Figure 5.1, information from "/Yahoo/Directory/Entertainment/Actors" and "/Google/Top/Arts/Movies/Actors and Actresses" would be retrieved. Red lines show a possible resulting set of alignments.

Semantic matching is a type of ontology matching technique [ES07] that relies on semantic information encoded in lightweight ontologies (a graph-like structure where the meaning of the nodes are unambiguous [GMZ07]) to identify nodes that are semantically related. For each node in the input graph-like structures, the algorithm tries to derive the concept that the node intends to represent. This is done by using not only the name or label for a given node, but also its position in the structure, exploiting implicit information encoded in the structure. After all the concepts have been computed, transforming the inputs into lightweight ontologies, the algorithm computes semantic relations (e.g., less general, more general, equivalence) between nodes from different input schemas.

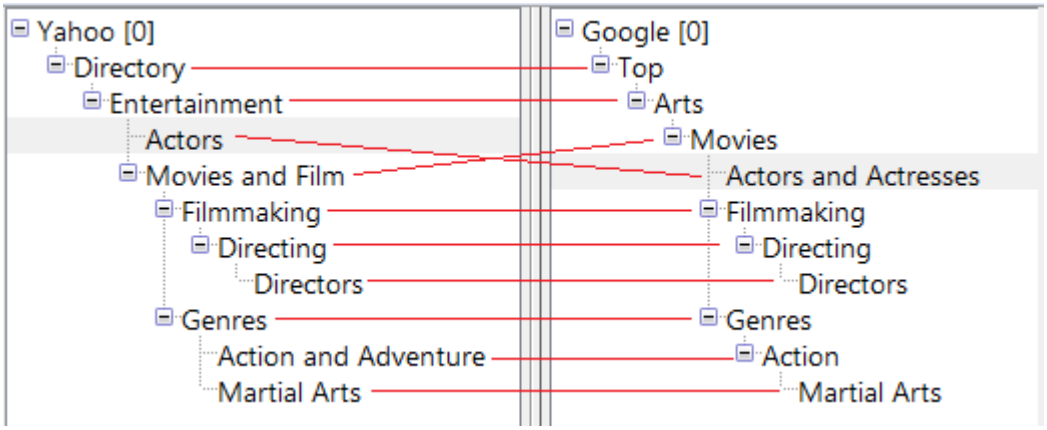


Figure 5.1: Two subsets of movies related Web directories from Google and Yahoo and an example matching solution

In many cases a set of structural properties need also to be preserved when computing semantic relations between input schemas. This is specially the case when matching web services or database schemas, where *i)* internal nodes, representing function symbols or table names, need to be matched to internal nodes and *ii)* leaf nodes, representing variable names or columns, need to be matched to leaf nodes. Furthermore, we need to return only a one to one mapping correspondences that will allow a direct translation between two services or database schemas. A similarity score is also computed to check whether the returned alignments can be considered good enough; however, the decision on whether the score is good enough is highly domain dependent, and therefore left as a user parameter.

The S-Match framework is open source implementation of the semantic matching algorithm. Given any two graph-like structures, the framework is able to transform them into lightweight ontologies and compute the semantic relations among the nodes. For example, applied to file systems it can identify that a folder labeled "car" is semantically equivalent to another folder "automobile" because they are synonyms in English, relying on the information store in a background knowledge.

There are different ways the set of mapping elements can be returned, according to the specifics of the problem where the semantic matching approach is being applied. In Section 5.2 we highlight the most important elements of the basic semantic matching algorithm as defined in [GS03]. Section 5.3 presents an adaptation of the basic semantic matching algorithm, called Structure Preserving Semantic Matching (SPSM), that is more suitable to be used when matching database schemas and web services. Section 5.4 presents the architecture of open source implementation of the algorithms, called S-Match. Section 5.5 discusses the related work. Finally, Section 5.6 summarizes the chapter.

The purpose of this Chapter is not to give a complete account for the internals of the semantic matching algorithms, but rather to give an overview in order to

understand why semantic matching and its basic components are important for achieving interoperability at the background by converting strings into semantic strings and schematic knowledge in order to compare entity types descriptions. These components are defined in Chapter 3 (Knowledge Components).

Acknowledgments: parts of this chapter were published in [GMY+08] in collaboration with Fausto Giunchiglia, Fiona McNeill, Mikalai Yatskevich, Paolo Besana and Pavel Shvaiko, in [MBPG09] with Fiona McNeill, Paolo Besana and Fausto Giunchiglia and in [GAP12] with Fausto Giunchiglia and Aliaksandr Autayeu.

5.2 The basic algorithm

The basic semantic matching algorithm was initially defined in [GS03] and extended in [GYS07]. The key idea is to find semantic relations (*equivalence* ($=$), *less general* ($\sqsubseteq$), *more general* ($\sqsupseteq$) and *disjointness* ($\perp$)) between the meanings (concepts) that the nodes of the input schemas represent, and not only the labels. The former refers to the set of data that would be contained under a given label, independently from its position; the latter refers to the set of data that would be contained under the given node and its position. The algorithm takes as input two graph like structures, converts them into lightweight ontologies and computes the set of alignments using a four steps approach, namely:

1. Compute concept of the label, for all the labels in the two input schemas.
2. Compute concept of the node, for all the nodes in the two input schemas, converting them into lightweight ontologies.
3. Compute existing relations between all the pair of labels from the two input lightweight ontologies.
4. Compute existing relations between all the pair of nodes from the two input lightweight ontologies.

5.2.1 From natural language to lightweight ontologies

In *step 1*, the algorithm computes the meaning of the label, considering its semantics. The main objective of this step is to unequivocally encode the intended meaning of the label, thus eliminating possible ambiguities introduced by the natural language such as, homonyms and synonyms. This is done transforming the natural language label into a complex concept (a set of concepts, from Model Object 6 with logical connectors).

This step can be divided into four sub-steps: tokenization, lemmatization, building atomic concepts (also known in the literature as Word Sense Disambiguations (WSD)) and building complex concepts.

Tokenization: Tokenization refers is the process of recognizing atomic words that are contained in the input label. For example, in the label "Action and Adventure" from Figure 5.1 the individual tokens are {"Action", "and", "Adventure"}. While the most common heuristic is to use the space as separator for recognizing the tokens, there are several cases when this is not sufficient. For example, labels from free-text annotation systems such as Delicious[1] do not allow spaces as word separators and, therefore, users just concatenate multi-words (javaisland) or concatenate them using the Camel case (javaIsland), slashes (java-island), underscores (java_island) or other separator they deem useful. Whenever possible, the above mentioned heuristics are applied. However, when not applicable, a dictionary-based tokenization can be performed, where the input label is tokenized looking for words that are part of the linguistic part of the Background Knowledge. For example, using the dictionary-based approach the label "javaisland" can be split into {"java", "island"} or {"java", "is", "land"} [AZP+10]. The output of this step is ranked to present the most plausible split first. The ranking prefers proposals with fewer number of splits and with the maximum number of tokens recognized by the dictionary.

Lemmatization: lemmatization to the process of reducing different forms of the same word into a single **base form**, as it would be found in a standard dictionary. There are a number of standard lemmatization heuristics taken from WSD. For example, transforming plurals to singular, "Directors" would be lemmatized to "director".

Disambiguation: In order to build atomic concepts, the algorithm relies on background knowledge presented in Section 3.2 which provides the possible senses for the computed lemmas given a context. Standard WSD techniques are applied in this step (see [IV98, Ste02] for a complete account on the WSD problem).

Build complex concept: finally, complex logical concepts are built using the previous information (for more details refer to [GS03]). The output of this step is a language independent Description Logic (DL) formula that encodes the meaning of the label.

Considering Figure 5.1 as an example, the senses that the label "Directors" could have are: "*manager of a company*", "*orchestra director*", "*theater director*" or "*film director*". Filtering the correct sense might seem like a trivial task for a human, but since there is no further information in the label itself, no further sense filtering can be done automatically during this step.

In *step 2*, the algorithm computes the meaning of the node; this is done by considering its position in the input graph-like structure, i.e., the path to the root.

[1] `http://delicious.com/`

In Figure 5.1, taking into account the position of the "Directors" node under the node "Filmmaking", the automatic sense filtering function would be capable of deriving the right sense: "*film director*". The output of this step is a lightweight ontology, i.e., a graph-like structure where the meaning of the nodes is unambiguous [GMZ07].

5.2.2 Node and tree matching

In *step 3*, the algorithm computes the relations between the concepts of label of two input lightweight ontologies. This is done with the aid of two types of element level matchers: semantic and syntactic. Semantic element level matchers rely on the conceptual part (see Section 3.2.2) to find semantic correspondences between the concepts of label. Instead, syntactic element level matchers, such as *N-Gram* and *Edit Distance* are used if no semantic relation could be found by the semantic element level matcher. Table 5.1 lists the available element level matchers.

Matcher Name	Type
Wordnet	Sense-based
Prefix	String-based
Suffix	String-based
N-gram	String-based
Edit distance	String-based

Table 5.1: Element level matchers

The output of this step is a matrix of relations between all atomic concepts encountered in the nodes of both *lightweight ontologies* given as inputs. This constitutes the theory and axioms which will be used in the following step (see [GYS07] for complete details).

In *step 4*, the algorithm computes the semantic relations ($=, \sqsupseteq, \sqsubseteq, \perp$) between the concepts at node. By relying on the axioms built on the previous step, the problem is reformulated as a propositional satisfiability (SAT) problem between each pair of nodes of the two input lightweight ontologies. This problem is then solved by a sound and complete SAT engine.

The output is a set of *mapping elements* in the form $\langle N_1^i, N_2^j, R \rangle$, namely a set of semantic correspondences between the nodes in the two lightweight ontologies given as input. Where N_1^i is the *i-th* node of the first lightweight ontology, N_2^j is the *j-th* node of the second lightweight ontology, and R is a semantic relation in ($=, \sqsupseteq, \sqsubseteq, \perp$). The lines between the nodes of the different trees in Figure 5.1 show a graphical representation of the mapping elements.

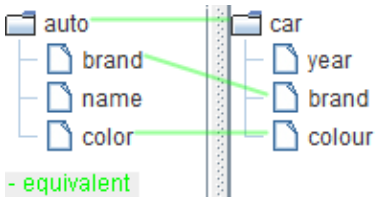


Figure 5.2: Result of Structure Preserving Semantic Matching (SPSM).

5.3 Structure Preserving Semantic Matching (SPSM)[2]

In many cases it is desirable to match structurally identical elements of both the source and the target parts of the input. This is specially the case when comparing signatures of functions such as web service descriptions or APIs or database schemas. Structure Preserving Semantic Matching (SPSM) [GMY+08] is a variant of the basic semantic matching algorithm outlined in Section 5.2. SPSM can be useful for facilitating, for instance, the process of automatic schema integration, returning a set of possible mappings between the elements of the schema. This can be used to compare definitions of entity types that were not created using the schematic knowledge presented in Section 3.3.

SPSM computes the set of mapping elements preserving the following structural properties of the inputs:

i) *only one mapping element per node is returned*. This is required in order to match only one element (a function, a schema name or attribute) in the first lightweight ontology, to only one element in the second lightweight ontology.

ii) *leaf nodes are matched to leaf nodes and internal nodes are matched to internal nodes*. The rationale behind this property is that a leaf node represents a parameter of a function (or an attribute in a schema, a column name in a database, or an attribute definition in a entity type), and an internal node corresponds to a function (or a class in a schema, or a table name in a database). By doing so, parameters containing values will not be aligned to functions.

Figure 5.2 shows the output of SPSM when matching two simple database schemas, consisting of one table each: table “auto” with columns “brand”, “name” and “color” on the left and table “car” with columns “year”, “brand” and “colour” on the right. Observing the results of SPSM in this example we can see that the set of structural properties is preserved:

i) The root node “auto” in the left tree has only one mapping to the node “car” in the right tree on the same level, that is, the root.

[2]Parts of this section were published in [GAP12] in collaboration with Fausto Giunchiglia and Aliaksandr Autayeu. and in [MBPG09] in collaboration with Fiona McNeill, Paolo Besana and Fausto Giunchiglia.

ii) The leaf node "brand" in the left tree is mapped to the leaf node "brand" in the right tree, and similarly with the leaf node "color" in the left tree and the node "colour" in the right tree — the leaf node is mapped to the node on the same level, that is, to a leaf node.

The output of SPSM is a set of alignments between the nodes of the input trees, and a numerical score in [0 1] indicating the degree of global similarity between them. This score allows us to detect not only perfect matches, which are unlikely to occur in an unconstrained domain, but also good enough matches. A match between two trees is considered to be good enough if this degree of global similarity exceeds some threshold value. Since the concept of good enough is very context dependent, in safety critical situation perhaps only a near-perfect match will be required but in other situations a much weaker match may suffice, this threshold can be set by the user according to the application domain [GMY$^+$08].

SPSM is performed in two steps: semantic node matching and structure preserving matching.

Semantic Node Matching: this computes all the possible set of correspondences between the nodes of the input lightweight ontologies. This is achieved by relying on the basic semantic matching algorithm outlined in Section 5.2.

Structure preserving matching: once we have the correspondences between the input nodes, the next step is to match the whole lightweight ontologies taking into account the set of structural properties we need to preserve, and determine the global similarity value between them. This step relies on the tree-edit distance algorithm that is designed to determine the cost of translating one tree into another through the application of three operations: (i) node deletion, (ii) node insertion, and (iii) node replacement [Tai79]. However, the original tree-edit distance algorithm does not consider the semantics behind these operations: for example, according to the standard algorithm, replacing the node "color" by "colour" in Figure 5.2 would cost the same as replacing it with "year", although it is clear that a the second replacement is less semantically meaningful than the first one.

In order to overcome this issue, the standard tree-edit distance algorithm has been enhanced so that the cost of performing each operation is dependent on the semantics of performing the change: that is, a smaller semantic change costs less than a large change. This is achieved by associating each edit distance operation to an operation from the theory of abstraction [GW92] and assigning a different value to each different abstraction operation. For example, replacing a node for another whose concept is semantically related (as defined in Model Object 7) costs less than replacing it by another with no semantic relation (for a complete account please refer to [GMY$^+$08]).

5.4 The S-Match open source framework

S-Match[3] is the Java open source implementation of the semantic matching algorithm presented in this chapter. It is currently distributed under GNU Library or Lesser General Public License (LGPL). Currently S-Match contains implementations of the basic semantic matching algorithm [GYS07], structure preserving semantic matching algorithm [GMY+08], as well as minimal semantic matching algorithm [GMA09]. S-Match is developed to have a modular architecture that allows easy extension and plug-in of ad-hoc components for specific tasks. Figure 5.3 shows the main components of the S-Match architecture, and a reference to the four steps of the Semantic Matching algorithm outlined in Section 5.2.

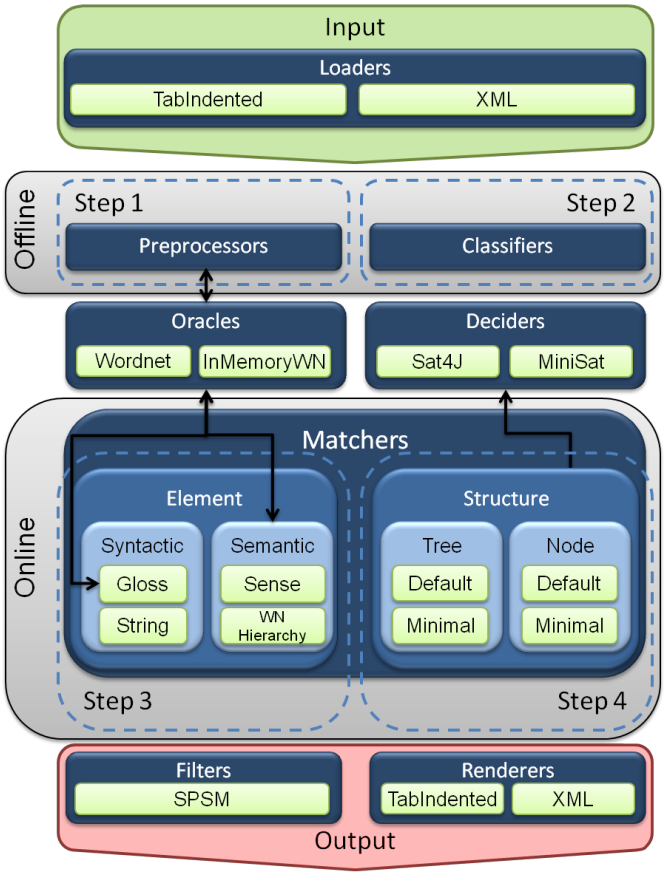


Figure 5.3: S-Match architecture

The *Loaders* package allows to load the input in various formats such as tab indented and XML formats. It also provides an interface for allowing to create custom loaders. The *Preprocessors* package provides the translation from Natural

[3] http://s-match.org/

language of the labels into concepts (step 1), using the *Oracles* package. The *Classifiers* package creates the concepts of the nodes in step 2 using the structure of the tree and the concepts of the label. These packages translate the input into lightweight ontologies. This translation needs to be performed only once, and can be done offline.

The *Matchers* package contains two sub-packages that perform steps 3 and 4 of the algorithm presented in Section 5.2, these are the *Element* and *Structure* packages respectively. The *Element* level matchers contain the semantic matchers that rely on the Oracle package to obtain relations between concepts, The syntactic matchers are, among others: gloss matcher, prefix, suffix, edit distance, NGram. The *Structure* level matchers are used to compute semantic relations between the concepts of the node relying on the *Deciders* package for solving the SAT problems.

5.5 Related work

A considerable amount of research has been done in the field of semantic matching. Different terminologies have been adopted to refer to the matching problem, such as alignment, merging, integration, mapping, and so on [KS03] in several combinations such as semantic matching [GS03, GYS07], ontology matching [ES07], ontology mapping [CSH06], schema matching [BBR11, BV11]. Even though each field has its own peculiarities, the general objective is similar, that is, to address the semantic heterogeneity problem at schema/structure level.

There are several comprehensive surveys regarding the matching problem in its related fields that were published during the years [KS03, DH05, SE05, CSH06, ES07] and more recently in [BBR11]. Therefore, we will not focus on creating yet another extensive literature review.

The Ontology Matching book [ES07] presents an exhaustive and detailed classification and state of the art survey. It provides a formalization of the ontology matching problem, classifying and evaluating algorithms according to various criteria. It provides a detailed description of the matching process and their applications. A more recent book, "Schema Matching and Mapping" [BBR11] presents the advances in the area in the last years, outlining several evaluation methodologies.

The Ontology Alignment Evaluation Initiative (OAEI)[4] aims at evaluating systems with a systematic benchmark containing several tracks such as expressive ontologies, directories and thesauri. The initiative has been active since 2005, and myself and Pavel Shvaiko have been in charge of running the directory track for 2008 [CEH+08], 2009 [EFH+09] and 2010 [EFM+10]. In total, 24 matching systems have participated in the directory track from 2005 to 2010 with increasing average f-measure. This shows the variety of research systems dedicated to addressing the matching problem and how their quality has been increasing over time.

[4] http://oaei.ontologymatching.org/

Some of these systems are Falcon[5], COMA++[6], Similarity Flooding[7], HMatch[8] and others.

Bernstein et al., [BMR11] enumerate several common techniques used by matching systems in order to compute the alignments. These are: *i)* linguistic matching (tokenization, stemming, lematization), *ii)* instance matching given the elements of the node, *iii)* structure matching considering the structure of the schema, *iv)* constrain-based matching considering the data types, values, etc., *v)* rules-based matching expressed in first-order logic, *vi)* graph-matching, *vii)* usage-based considering the logs of the queries, *viii)* document content based matching normally using TI-IDF (term frequency times inverse document frequency) and *ix)* the use of auxiliary information such as a linguistic Oracle or background knowledge. The work presented in this Chapter uses *i), iii), v) and ix)*.

Several systems also use a combination of the above mentioned techniques; some examples are OLA [FKEV07] and COMA++. The assumption is that no basic matcher can solve efficiently all the variety of matching tasks and by using them in combination, the resulting matching solution can have an higher performance. However, even when combining several techniques, there is still no automatic solution for the matching problem with perfect recall. The authors in [ACMT08] state that only 80% of the existing alignments can be automatically found, and the remaining 20% have to be solved by experts. These estimations are consistent with six years of evaluation of 24 systems at the OAEI evaluation campaign (2005 to 2010), where the highest recall for the directory track was 84% (obtained by OLA in 2007 [KEV07]).

In order to further increase the quality of the results, several systems include a manual step (the user-in-the-loop approach) for providing the missing alignments and/or validating the results of the automatic approaches [ACMT08, JMSK09]. The approaches outlined by Zhdanova and Shvaiko [ZS06] and McCann et al.[MSD08] take the manual validation a step further distributing the effort in order to minimize the workload of experts. Several issues that need to be addressed in this "crowdsourcing"-based approaches are how to compute agreement, if a result obtained in one domain is still valid for other domains, how to incentivize people to participate and the trustworthiness of the validators.

5.6 Summary

In this chapter we have presented the semantic matching approach and an architecture of the open source implementation, S-Match. We outlined the important components that allows the conversion of natural language labels into a formal representation that removes the ambiguities of the natural language. This process

[5] ws.nju.edu.cn/falcon-ao

[6] dbs.uni-leipzig.de/Research/coma.html

[7] www-db.stanford.edu/~melnik/mm/sfa/

[8] islab.dico.unimi.it/hmatch/

relies on a linguistic oracle in the form of a background knowledge as presented in Section 3.2.

The Structure Preserving Semantic Matching (SPSM) extends the basic semantic matching algorithm by introducing constraints that allow to preserve structural properties when matching graph-like input schemas. This allows us to determine fully how structured terms, such as web service calls or entity types definitions, are related to one another, providing also a similarity score based on the theory of abstraction.

The purpose of this Chapter is not to give a complete account for the internals of the semantic matching algorithms, but rather to give an overview in order to understand why semantic matching and its basic components are important for achieving interoperability at the background and schematic knowledge levels presented in Chapter 3 (Knowledge Components).

The approaches presented in this Chapter rely on background knowledge to derive semantic relations, and therefore, it is dependent on their coverage in order to produce meaningful results. In Chapter 6 (Background Knowledge Evolution) we will see how to improve the coverage of the background knowledge in highly dynamic domains, such as computer science, where the time lag between the creation of new terms and their addition to the background knowledge by experts can be significant (the addition of new words to dictionaries is a process that normally take years).

Chapter 6

Background Knowledge Evolution

6.1 Introduction

As knowledge is in constant evolution, new concepts and words may arise to denote new or existing ideas and or to describe entities. In Chapter 3 (Knowledge Components) we see how these concepts and words are encoded in our Background Knowledge (BK). In Chapter 5 (Semantic Matching) we see how this Background Knowledge is essential in the Word Sense Disambiguation (WSD) process for removing the ambiguity of natural language and formalizing the meaning of strings in the system.

The dependency on the Background Knowledge for deriving the formal semantics is, however, a problem in highly dynamic domains, such as computer science. As shown in [APZ11a], a Knowledge Organization System (KOS) such as WordNet [Fel98] (from where we can initially populate the BK) only covers between 49% and 71% of the terms used in folksonomies for the "Computer Science" and "Education, Travel, Cooking" domains respectively. The time lag between the creation of new terms in dynamic domains and their addition to the Background Knowledge by domain experts can sometimes be significant (the addition of new words to dictionaries is a process that normally take years).

Two base cases need to be addressed when adding new elements to the background knowledge.

1. the word is recognized by the linguistic part of the BK, but lacks the intended meaning in the conceptual part. For example, while "ajax" and "mac" are both recognized in Wordnet, the only available sense for "ajax" is "*a mythical Greek hero*" and for "mac, machintosh, mackintosh, mack" is "*a waterproof raincoat made of rubberized fabric*". Clearly, the senses for the Computer Science domain are missing.

2. the word is not recognized by the linguistic part, which has the following sub

cases:

- the word is not significant (e.g., a proposition), and therefore should be ignored
- the word denotes an existing concept in the conceptual part of the background knowledge, therefore we need to add the new word to the corresponding synset;
- the word denotes an unknown concept, therefore, the concept, word and synset have to be created. Additionally, when creating the concept we need to create the concept in relation to already known concepts (i.e., define its meaning in relation to what it is already known).

In this Chapter we propose an approach for computing these missing senses in order to improve the coverage of the background knowledge. We try to minimize the user involvement whenever possible, assuming that the user attention is a scarce resource and thus should be used only when absolutely necessary. As noted in [APZ09], when in presence of a new term or concept we could adopt the following approaches for updating the Background Knowledge:

- Automatically update. We define a sense induction algorithm presented in Section 6.4.2 (initially described in [AZP+10]) that works with statistical information using clustering techniques. If the extracted concept has a certainty over a given threshold, then this new concept and its corresponding words and synset could be automatically added to the background knowledge.
- Semi-automatically. The results of automatic algorithms could be shown to the users, asking the user for his/her approval. Game mechanics [vA06] and crowd-sourcing [vAMM+08] could be employed in order to distribute this task and make it more appealing for the end user.
- Manually, by asking the user to create the concept and provide the necessary information for creating the words and synsets. The workload can be split following a collaborative approach [BSW+07].

We define a sense induction algorithm that differs from the state of the art by being able to recognize homographs, i.e., that one word can have many senses and compute the correct sense for which it is begin used. We use a folksonomy based systems, Delicious[1], as a use case scenario and define a model where the annotations provided in the folksonomy are converted into semantic strings (see Model Object 5). However, due to the lack of benchmark datasets and agreed-upon evaluation methodology, we create a manually validated dataset and propose an evaluation methodology (presented in Chapter 11).

[1] `http://delicious.com/`

The remainder of this Chapter is organized as follows: Section 6.2 introduces the use case that we will use to model and later evaluate the sense induction algorithm; Section 6.3 presents the related work; Section 6.4 defines the sense induction algorithm and the clustering distances needed by the sense induction algorithm are presented in Section 6.5; Finally, Section 6.6 summarizes the chapter.

Acknowledgment: The work presented in this chapter has been carried out with Pierre Andrews and Ilya Zaihrayeu. Parts of this Chapter have been published in [APZ11a] and [APZ11b].

6.2 Motivating example: Folksonomies

Folksonomies are uncontrolled Knowledge Organisation Systems (KOS) where users can use free-text tags to annotate resources [Wal07], this is also known as collaborative tagging [GH06]. The main characteristic of the folksonomy (folk + taxonomy) is the bottom-up approach for reaching consensus on emergent ideas, without any a-priory agreement on the vocabulary by users. The main advantage is the distribution of the tagging workload among all the users. This, in turn, lowers the barrier of user participation since one user can see and adopt the tags assigned to a resource by other users, thus simplifying the tagging process, or even more importantly, can search and discover new resources that were tagged with the same tags as the user's tags [AN07].

Folksonomies create a network of user-tag-resource triplets that encodes the knowledge of users [Mik05] (see equation 6.1). The model is widely adopted in systems such as Delicious, Flickr[2], Diigo[3], Bibsonomy[4] and many more, mainly due to its simplicity. However, because they are based on the use of free-text tags, folksonomies are prone to language ambiguity issues as there is no formalisation of polysemy/homography (where one tag can have multiple senses) and synonymy (where multiple tags can have the same sense) [GH06]. This lack of explicit semantics makes it difficult for computer algorithms to leverage the whole knowledge provided by the folksonomy model. For instance, in [APZ11a] we show that exploiting these explicit semantics can improve search results.

6.2.1 Syntactic Folksonomy

Mika [Mik05] introduces a formalisation of the folksonomy model to ease its processing using multimodal graph analysis. Doing so, the author enables the formal representation of the social network resulting from the folksonomy building activity. It represents a folksonomy as a tripartite graph composed of three disjoint types of vertices, the *actors* A (the user creating the tag annotation), the *concepts* C (tags, keywords) used as metadata and the *objects* O or resources being annotated.

[2] `http://www.flickr.com/`
[3] `http://www.diigo.com/`
[4] `http://www.bibsonomy.org/`

A *free-text tag annotation* is thus a triple combining the three vertices as defined by Equation (6.1).

$$T = \langle u, t, r \rangle \text{ where } u \in A, t \in C \text{ and } r \in O \tag{6.1}$$

According to Mika, such tripartite graph can be used to describe an ontology representing the knowledge of the community that created this folksonomy. This model has been used since to exploit different social networking analysis tools and distributional semantic models to extract a more formal representation of the semantics encoded in these tripartite graphs (see [GSCAGP10] for a review).

6.2.2 Semantic Folksonomy

An important point in Mika's [Mik05] description of the folksonomy model is that "tags" or "keywords" are considered to be mapped one-to-one to the concepts of the ontology and that these (the tags or keywords) are the basic semantic units of the language used in the community that created the folksonomy. However, we believe that we need to separate the linguistic part from the conceptual part to properly encode the semantics in the folksonomies. This will enable a better understanding of its underlying semantic and of the overlap of vocabularies between the users of the folksonomy.

In fact, tags and keywords, while they represent a specific concept and have a known semantic for the agent that creates them, are just stored and shared in the folksonomy as purely free-form natural language text. Because of the ambiguous nature of natural language, a number of issues such as synonymy, homography, base form variation and specificity gap arise when sharing only the textual version of the annotations (see Section 3.2 for a more detailed account of the problems with natural language).

Indeed, as we show in our evaluation (see Section 11.2.2), such issues are present in a real application based on the folksonomy model, in our case study, Delicious. We thus propose to extend the free-text tag annotation model in equation 6.1 by replacing the tag t string by our model of semantic string presented in Model Object 5 in Section 3.2.1. Equation 6.2 shows the semantic annotation. Where the semantic string $string_{sem} = \langle t, \{w\_s\} \rangle$ contains the original tag t, and a set of word senses $w\_s = \langle word, c \rangle$, each containing a token (a $word$ in t in its base form) and the concept c it represents (see Model Object 4 and Model Object 5).

$$T = \langle u, string_{sem}, r \rangle \tag{6.2}$$

The "semantification" of the tag is a process similar to that of Word Sense Disambiguation (WSD) (see Section 5.2.1: "From natural language to lightweight ontologies"). However, while there are existing WSD algorithms in the state of the art, they are not completely adapted to folksonomies. WSD algorithms use an existing vocabulary to link terms (in our case tags) to concepts, thus discovering

the semantics of the tags used. However, as shown in [APZ11a], WordNet only covers between 49% and 71% of the terms used by the users of the folksonomy. That is, WSD cannot be applied to up to 51% of the tags in the folksonomy as they do not have any sense in the background knowledge, if based in Wordnet.

While the computer cannot understand the meaning of the free-text tags used (either because the word or the concept is missing in the background knowledge), the users always know the meaning they wanted to use when they tagged a resource. So, if they tagged a resource with "atm", in their mind, at the time of tagging, they knew exactly if they meant the "*cash machine*" or the "*a standard unit of pressure, the atmosphere*". This principle has already been widely illustrated in the automatic ontology building field where machine learning is used to extract the so-called "emergent semantics" [AMO+04].

6.2.3 Semantifying Delicious

To analyze our semantified folksonomy model we use Delicious, a simple folksonomy that allows users to tag Web pages (the resources r) using free-text annotations. We obtained the initial data from the authors of [WZB08] who crawled Delicious between December 2007 and April 2008. The aim of using this dataset is two-fold:

First: we want to convert the free-text annotations to semantic annotations. This process allows us to evaluate WSD algorithms and the coverage of the Background Knowledge.

Second: given the lack of gold standard dataset for evaluating the sense induction algorithm, we manually build and evaluate the conversion from free-text to semantic strings, which will serve as the gold-standard dataset not only for the sense induction algorithm, but also for other future WSD algorithms. The dataset is currently published following the Linked Open Data principles at `http://disi.unitn.it/~knowdive/dataset/delicious/`.

The dataset and the results are presented in Chapter 11.

6.3 Related work

The work by [GSCAGP10] provides a good survey of the field of semantic disambiguation in folksonomies. The authors present a *unified process for the association of semantics to tags* in four steps:

Data selection and cleaning includes the identification of the folksonomy dataset to be used. Common examples are Delicious and Flickr. Once selected, the dataset has to be cleaned out of undesired content, e.g., spam tags.

Context identification refers to the extra information that is to be used in order to aid in the selection of the meaning of the tag. For example, in free text processing [AM02a], the context can be the whole sentence, exploiting structural information. This information is not available in folksonomies, as tags are not placed in grammatical sentences, however, different contextual information can be derived from the folksonomy structure.

Disambiguation (WSD) tries to address the polysemy issue due to the natural language used in tags. This step tries to separate different uses of the homonymous or polysemous tags based on the context in which they are used.

Semantic identification tries to link the sense of the tag to a formal meaning given by a Knowledge Organization System (KOS). This KOS can be, for example, an ontology like DBpedia [ABK+08] or a thesaurus such as Wordnet [Fel98].

This unified process for the association of semantics to tags makes the assumption that the target formal Knowledge Organization System (KOS) (e.g., Ontology, Thesaurus, CV) already contains the concepts/classes and entities/instances that are to be linked to the folksonomy tags. However, as pointed out by [APZ11a], the coverage of the KOS, in the particular case of Wordnet 3.0, is only around 49% for the Delicious folksonomy in the domain of Computer Science. This means that, even if we achieve a perfect precision and recall in the process of assigning semantics to tags as described by the WSD step by [GSCAGP10], we will only be able to assign formal semantics to as much as half of the tags created by the users of the folksonomy. This language coverage issue raises the need for a new step in the "semantification" process of tags in folksonomies: *sense induction*.

Sense induction is the task of discovering automatically the intended meaning of a possible ambiguous word in a given context. Differently from the disambiguation task, where the senses are assumed to be defined by a KOS, sense induction tries to overcome this limitation by also dealing with missing senses in the KOS [BL09]. Therefore, besides considering the polysemy issue, sense induction has to deal with the maintenance of the concept with respect to the KOS.

This maintenance, also known as ontology evolution, is defined by [FMK+08] as "*the process of modifying an ontology in response to a certain change in the domain or its conceptualization*". However, the ontology evolution literature is often defined in terms of single-user environments directed to aid experts in the manual maintenance of the ontology and does not take into account important characteristics of collaborative schemas and shared KOS [SMMS02, HHSS05]. Some of the approaches cited by [GSCAGP10] do consider the creation of the senses from the context in which the tag resides, for instance [Mik05, SM07].

Considering community-built KOS, [BSW+07] proposed an ontology maturing process by which new knowledge is formalized in a bottom-up fashion going through several maturity stages: from informal emergent ideas using input from social annotation systems such as folksonomies, to formal lightweight ontologies

via a learning process involving users. This process also relies on manual user effort to evolve the KOS.

Ontology learning [Zho07] use machine learning techniques for automatically discovering and creating ontological knowledge. The automatic process is able to discover knowledge faster and at larger scale than manual approaches. Ontology learning can create the ontology from scratch [Mik05, BSW+07], or refine and expand an existing ontology [RMT08, HHSS05]. The author identifies three types of learning mechanism; *i)* statistics based, *ii)* rule based and *iii)* hybrid. [AMO+04] proposed several generic characteristics and issues that any system for *emergent semantics* should consider and address.

One method used to automatically extract the semantics from folksonomies is what is called *tag clustering* and its principle is based on machine learning clustering algorithms [XW05, Sch06]. This clustering is based on the principle that similar tags will have the same meaning and can thus be attached to the same *"concept"* in the created vocabulary. For instance, if the algorithm finds out that "opposition" and "resistance" are similar, then it can associate it to one concept for that meaning. One of the main issues is thus to compute the similarity between tags to run the clustering algorithms that will attach similar tags together. To do this, all the methods available currently use a mix of measures based on the collocation of tags on resources and their use by users. If two tags are often used by the same user on different resources or by different users on the same resource, then they can be considered similar [GSCAGP10].

An important point in Mika's [Mik05] description of the folksonomy model is that "tags" are considered to be mapped one-to-one to the *concepts* of the ontology and that these are the semantic units of the language used in the community that created the folksonomy. In our opinion, this assumption, which is often found in the state of the art, is one of the first weak points of these approaches as it makes the assumption that one tag can only have one meaning [Mik05, MDA08]. Thus these algorithms can find synonyms of the most popular sense but cannot deal with the polysemy/homography of the tags. For example, if the tag "java" is collocated with "indonesian island" on 200 resources and with "programming language" on 1000 resources, then it will be considered to be similar to the latter and the fact that it has a second meaning is lost. However, [ZWY06] show that tags are often ambiguous in folksonomies (their study is also based on Delicious[5]) and can bare more than one meaning.

Another important issue raised by [GSCAGP10] is the lack of a common evaluation methodology and dataset. In fact, as we will discuss in Chapter 11 (Evaluation of Background Knowledge Evolution), there is an issue in the state of the art as many papers only report anachronistic evaluations that cannot be reproduced or compared, e.g., [LDZ09, GSSAC09, VHS07, SM07].

[5] `http://www.delicious.com`

6.4 Sense Induction

Many approaches are described in the state of the art and [GSCAGP10] tries to formalise the problem in a number of general steps to be achieved to resolve the semantics of tags in folksonomies. In this section we discuss a generic approach that is inspired by the state of the art and tries to also tackle the issues discussed earlier, such as the identification of homographs before identifying synonymous. As in most of the state of the art approaches, a prior step to assigning a sense to the tags is to clean them; we thus first describe an accurate preprocessing task before going on to the core algorithm.

6.4.1 Preprocessing Tags

An issue with the most popular folksonomies available at the time of writing is that many of the tags are written without spaces or without following a standard lemmatised form, so tags such as "javaisland", "java_island" or "hike", "hiking", "greathikes" are often found and cannot be easily matched together or to an existing controlled vocabulary, in our case, the linguistic part of the Background Knowledge. In a sense, this is not an issue of the folksonomy model but an issue of the user interfaces offered by the main tagging services such as Delicious or Flickr. In fact, these systems did not allow the use of tags with spaces and thus the users had to find alternative ways for creating multi-word tags. The issue of base form variation in addition, is a usual issue when dealing with natural language.

As in standard Natural Language Processing (NLP), where free text is first tokenised and lemmatised, we need to split and normalise the tags from the folksonomy (see Section 5.2.1: "From natural language to lightweight ontologies"). However, as noted by Andrews [AZP+10], standard tokenisation is based on puctuation and spacing to extract tokens, in our context, this approach cannot be used as the words that need to be tokenised do not include space or punctuation (in most cases) within the tag (e.g. javaisland). For lemmatisation, we can use a similar approach to the standard NLP approaches, testing for common derivations of words or verbs or special derivation of specific words[6]. However, there are cases where lemmatisation and tokenisation have to be performed together, for instance "teachersresource" has to be tokenised and lemmatised at the same time as the first token "teachers" is a variation of the dictionary form "teacher".

To perform this task we use the algorithm defined by Andrews in [AZP+10] based on a standard dictionary search approach, where we compile the vocabulary (lemmas) from the linguistic part of the BK, preloaded with WordNet 3.0, in a Final State Automata (FSA). The output of this step is a list of possible semantic strings (see Model Object 5 in page 17), each corresponding to a possible "split". The words in the outpout can be linked to a concept, or marked as unknown. The list is ranked first by the number of splits (the fewer the better) and then by the number

[6] which are provided by thesaurus such as WordNet

of unknown words (the fewer the better too). This approach proved an accuracy of 80.4%, as each split was manually validated by the users.

6.4.2 Algorithms

Once the tags have been preprocessed in tokens and lemmas, we have to perform the "disambiguation" and "semantic identification" steps introduced by [GSCAGP10] to find the semantics of each token. Many different techniques are used in the state of the art, but the general consensus is that a form of clustering can be used to group tags together to form senses. In this section we propose to generalize different aspects of approaches found in the state of the art and to introduce some slight improvements. The proposed algorithm is then used in Section 11.3 to provide a formal evaluation of different variations of the approach that we believe simulate the techniques found in the state of the art.

The method used to extract the semantics from folksonomies is what is called *tag clustering* and its principle is based on machine learning clustering algorithms [XW05]. This clustering is based on the principle that similar tags will have the same meaning and can thus be attached to the same *"concept"* in the created vocabulary. For instance, if the algorithm finds out that "opposition" and "resistance" are similar, then it can associate it to one concept for that meaning. One of the main issues is thus to compute the similarity between tags to run the clustering algorithms that will attach similar tags together. To do this, all the methods available currently use a mix of measures based on the collocation of tags on resources and their use by users. If two tags are often used by the same user on different resources or by different users on the same resource, then they can be considered similar [GSCAGP10].

This assumption on the computation of the similarity of tags is, in our opinion, one of the first weak points of these approaches as it makes the assumption that one tag can only have one meaning. Thus these algorithms can find synonyms of the most popular sense but cannot deal with the polysemy of the words. For example, if the tag "java" is collocated with "indonesian island" on 200 resources and with "programming language" on 1000 resources, then it will be considered to be similar to the latter and the fact that it has a second meaning is lost. However, [ZWY06] show that tags are often ambiguous in folksonomies (their study is also based on Delicious[7]) and can bare more than one meaning. In the algorithm we propose, we add an extra step to the clustering to first identify the diverse senses of polysemous tags and in the following clustering steps, we do not consider tags directly, but the unique senses that they can take.

We propose to adopt a parametric based clustering approach slightly different from the standard KMeans and KNN algorithms that are often discussed in the state of the art of ontology construction from folksonomy (see, for a review [GSCAGP10]). In fact, these algorithms, while being the most popular in the clustering field, are not well tailored to our application domain as they take as an input-parameter the

[7] `http://www.delicious.com`

number of expected clusters (the K in the name). The state of the art approaches on ontology building from folksonomies cluster all the tags together to find all the concepts that they represent (see figures two and four in the review of Garcia-Silva et al. [GSCAGP10]). In this case, they can optimise the K parameter to find the best overall number of clusters for their dataset. However, in our approach, we have added an extra step where clustering is applied to detect the different senses in which one tag can be used. In this case, we cannot find an overall optimal value for the number of clusters to look for as each term might have a different number of senses.

Thus, we need to use a clustering algorithm that can work without this parameter as input. We use the DBScan algorithm [EpKSX96] to do a density based clustering. This approach to clustering has various advantages for our application:

- it does not require as input the number of clusters to be found. Instead it takes two parameters: λ, the minimum distance between two items to put them in the same cluster and m the minimum number of items in a cluster.

  λ is easier to optimize in our use case than to compute the K parameter as we can find it by studying the accuracy of each clustering step.

- while the KMean and KNN algorithms assign all items in the clustering space to a cluster, the DBScan algorithm can decide that some of the items to be clustered are noise and should not be considered. This is very important in our application domain as it allows for leaving out very personal or subjective uses of a term that might not be aligned with the rest of the community understanding of the term; and

- the DBScan algorithm can detect clusters that have more complex "shapes" than the standard hyperspherical clusters returned by vector quantization based clustering such as the KMeans and KNN [XW05].

While there is already some research done on diverse similarity measures applicable to concept detection and learning in the Natural Language Processing field (for instance [AM02a] or [Jam09]), the existing clustering techniques discussed in the folksonomy field are only considering raw collocation counts (of tags, resources or users) as a similarity measure between tags. For instance, [AM02a] proposes to combine four different measures to compute sense similarities: the *topic signature*, the *subject signature*, the *object signature* and the *modifier signature*. While most of these measures can only be applied to textual documents as they require to know noun-verb relationships in a sentence, the *topic signature* is interesting in the domain of folksonomy where one of the only context we have for computing the distances is the list of collocations. However, these collocations can be considered and weighted in different ways and [Jam09] points out that simple vector distances or cosinus distances between *topic signatures* are not always powerful enough. The authors show that information based measures –

such as the Kullback-Leibler divergence of word distribution, the mutual information – can be used to have more powerful measures of semantic distances between concepts based on the Distributional Semantics principles [Lin98]. The authors of [WSyVZ08] have proven that this measure can be applied with success to the domain of folksonomies to disambiguate tag senses.

For the algorithm that we discuss in this section we use clustering algorithms relying on distance measures between User-Resource pair and between tag senses. We are currently experimenting with different measures, from the standard tag collocation measures proposed in the current state of the art to the more advanced distributional measures described above.

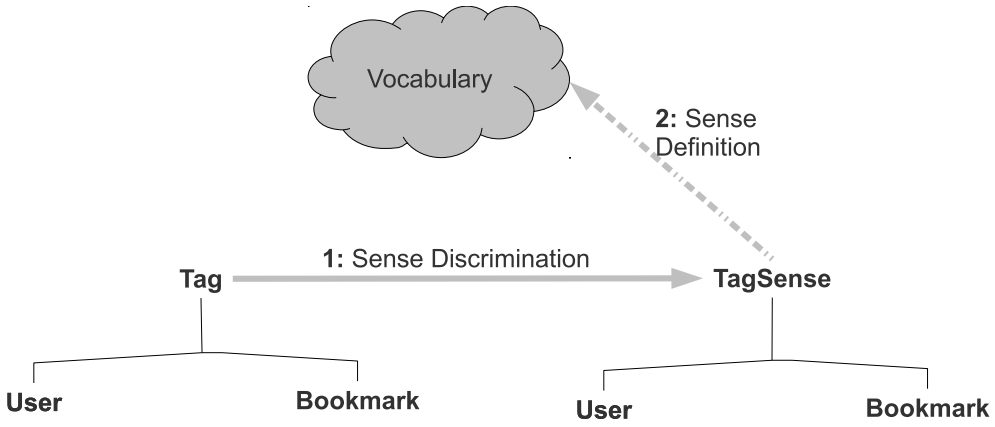


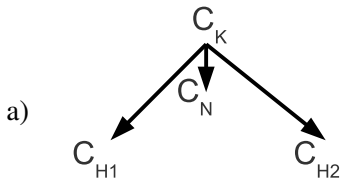
Figure 6.1: Sense-User-Bookmark Tripartite graph

To enrich the structured vocabulary with a new concept from a free-text tag, we propose to do the concept detection in three stages:

1. For each tag, we cluster the user-resource bipartite graph that are attached to this tag. By doing so, as was hinted by [AGH07], we discover the different meanings of the tag. By considering each cluster to be a tag sense, we replace the tag in the user-resource-tag tripartite graph by its senses and the tripartite graph becomes a user-resource-sense graph as illustrated in Figure 6.1. In this way, if we consider our previous example, the tag "java" will be split in two senses: `java-1`, similar to "indonesian island" and `java-2`, similar to "programming language".

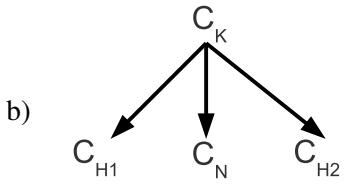
2. We then apply the same principle as the one discussed in the state of the art on the user-resource-sense tripartite graph to cluster similar senses together (see [GSCAGP10] for a review).

3. Once the tag senses have been clustered together, we identify new concepts for each of the clusters. This process is equivalent to finding the relation (in particular hypernym/hyponym relations) of the new concept (represented by the cluster of tag senses) in the structured vocabulary. This can be achieved



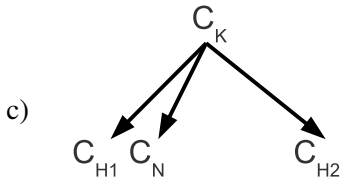
$D_{N-K} \leq D_{N-H}$
$D_{N-K} < \frac{1}{\rho} \times D_{K-H}$

C_K and C_N are "synonymous"



$D_{N-K} \leq D_{N-H}$
$D_{N-K} \geq \frac{1}{\rho} \times D_{K-H}$

C_K is more general of C_N



$D_{N-K} > D_{N-H}$

C_N will be compared to C_{H1}, recuresivelly applying steps a), b) and c)

Figure 6.2: Decisions to Extend the Concept Taxonomy

by applying a hierarchical classification approach similar to the one proposed in [AM02a]. In their approach to ontology building, they consider a similarity measure between a *known* concept C_k in the vocabulary and a new concept C_n.

- If the distance between these two concepts is smaller than the distance between C_n and any of the hyponyms of C_k, then C_n is considered to be the hyponym of C_k.
- Otherwise, they continue the search down the conceptual hierarchy.

We alter this approach by splitting it in three cases as we believe that there can also be cases in which the new concepts C_n are actually synonyms of an existing concept C_k. The updated solution, summarized in Figure 6.2, is as follows:

- if the new concept C_n is closer to the existing concept C_k than to any of the its hyponyms, but much more – this is defined by the parameter ρ as defined in Figure 6.2a) – similar to C_k than any of its hyponyms, then it is most likely that C_n is a synonym of C_k (Figure 6.2a)[8]);
- if the new concept C_n is closer to the existing concept C_k than to any of the its hyponyms, but not much more similar to C_k than any of

[8]where D_{i-j} is the distance between C_i and C_j.

its hyponyms, then it is most likely that C_n is more specific than C_k (Figure 6.2b));

- if the new concept C_n is closer to the a hyponyms of C_k than C_k, then we recursively apply these three steps to this most similar hyponym (Figure 6.2c));

We apply this search procedure on our in the conceptual part of the Background Knowledge (initially populated with WordNet), starting from the root of its conceptual *is-a* hierarchy.

This approach is parametric as it depends on the value of ρ, which specifies the threshold to decide if a new concept is more specific than an existing concept or is just a synonymous. This parameter will be different depending on the specific application domain and will decide how much specific the structured vocabulary will get.

6.5 Distance measures

In the previous section, we have described an algorithm for sense induction, this algorithm can be divided in three main steps that all require the computation of distances to work. The two first steps are based on standard clustering techniques that require to compute distances between the elements to be clustered (i.e. the tags to find sense clusters or the sense clusters to find synonymous tags); the third step uses the distance between concepts to place them in the conceptual part of the Background Knowledge.

We have not discussed the particulars of the distances to use in the previous section as many kind of distances can be used to perform the same task. In this section we mention a set of intuitive collocation distances that we have found across the state of the art of sense disambiguation and induction within folksonomy.

Note that there are interesting works in sense induction from free text, for example [Jam09, AM02a] and in graph clustering [Sch07] that could be ported to compute smarter distances. For instance, [WSVZ08] use an interesting probabilistic framework to find tags that would help disambiguate an existing set of tags. What [WSVZ08] are trying to do is, given two similar sets of tags, to find a pair of new tags to add to each set to make them as different as possible. For example the tag "Cambridge" could refer to different places in the world, but the sets of tags "Cambridge, UK" and "Cambridge, Massachusetts", while they share a tag, are unambiguous. While the application is somehow different, a similar approach could be used to build an annotation distance that would encode more information about tags than pure collocation. However, this is not in the scope of this Chapter and here we will limit ourselves to introducing the existing state of the art distances and later in Chapter 11 evaluate their performances.

In the first step of the algorithm described in Section 6.4.2, we try to cluster different annotations of resources with the same tag to find out different senses of

this tag. To do this, we need to compute the distance between annotations (i.e. a user-resource-tag triplet as defined in eq. 6.1). In the second step of the algorithm, the clustering is applied on sets of annotations, representing tag-senses that we try to cluster together to find synonyms (see Figure 6.1). To do this, we need to compute the similarity between sets of annotations, which can be seen as a function of the distances between the individual annotations in each set. In the final step, we are not dealing with clustering, but are trying to find the distance between a new concept (and the set of annotations where it is used) and an existing concept (and the set of annotations where it is used). That is, we are again comparing sets of annotations, thus trying to find the distance between two annotations.

While the best distance for each step might be different, we can see that the principle is the same and that we need to find the best way to compare two annotations. An annotation is a triplet representing a tag annotation on a resource by a particular user. We can thus compare annotations on three main dimensions: the tag used, the resource being annotated or the user doing the annotation. A distance between annotations can thus be defined as:

- **Tag Collocation**: The number of collocated tags on the resources of two annotations.
- **User Collocation**: The number of users that annotated the resources of two annotations.
- **Resource Collocation**: The number of resources two users share.
- **Tag-Tag Collocation**: The number of resources that are annotated by two tags.

We implement some of these collocation-based distance measures and report the findings in Chapter 11. However, note that the sense induction algorithm presented in Section 6.4.2 is independent of the exact distance used, and further improvements can be done defining better performing distances in the future.

6.6 Summary

In this Chapter we propose a parametric algorithm for inducing new senses and update the linguistic and conceptual part of the background knowledge to address the dynamic nature of knowledge and to allow new concepts and terms to emerge in a bottom-up fashion. The algorithm is based on the usage of terms in collaborative annotation scenarios such as folksonomies. The advantage of this approach is the distribution of the workload among all participants of the distributed system, and the fact that users can reuse the annotations created by other users.

The novelty of the three-step sense induction algorithm resides in the new step introduced at the beginning were we first cluster annotations (tags) that are used in a similar manner to create tag senses. By doing so, we can capture the different

senses of the words used in the tags (thus addressing polysemy) and later in the second step we apply standard clustering techniques to group these tag senses together to create sets of synonyms tag senses. Each tag sense synonym set is considered to represent a concept, and thus, in the third and final step of the sense induction algorithm we compute the relation of this new concept with respect to all the other already existing concepts in the conceptual part of the Background knowledge.

The three steps of the sense induction algorithm rely on the computation of distance measures between annotations for creating the clusters and compute the position of the new concept in the conceptual part. We presented some basic distance measures based on collocation used in the state of the art, however, given the fact that the algorithm does not depend on any specific type of distance measure, new distance measures can be tested and adopted from other state of the art approaches with no change on the basic sense induction algorithm.

In Chapter 11 "Evaluation of Background Knowledge Evolution" we propose an evaluation mechanism. In order to perform this evaluation, we first create a manually validated gold standard dataset given the lack of agreed upon evaluation mechanisms and publicly available evaluation datasets.

Chapter 7

Knowledge propagation

7.1 Introduction

In a distributed system where users have their own knowledge, each user can define new local knowledge (words, concepts, entity types, entities) and make it available so that others can benefit from it. The issue arises when this "new" knowledge is not understood by other users, therefore, a process of propagation of knowledge need to be performed, and learning is needed for the users that acquire this "new" knowledge.

Note that by user here we mean an agentive user, which can be a person, or an organization (as defined in the Personal layer in Chapter 4). In the case of organizations, given that it normally represents a group of people, we will normally refer to them as communities (as defined in the Community Layer in Section 4.3). In this case, a community will normally be centered around one (or some) areas of interest or domains. There can be propagation of knowledge between users and communities in both ways, and also among users and among communities. In the rest of the document when describing the problem of knowledge propagation and the possible approaches we will use the term "user" as both the user as person and communities. This is because as we saw in Chapter 4, both layers have the same knowledge structure based on what is defined in Chapter 3: "Knowledge Components", therefore the same process of propagation of knowledge applies between them.

The process of propagation of the knowledge can normally be activated when:

1. initially bootstrapping the local knowledge of a user importing the relevant part of the Universal Knowledge (UK),

2. importing knowledge locally when becoming part of a community,

3. copying knowledge from other users and/or communities.

As discussed in Chapter 4, we distinguish three layers of knowledge, Universal (or global), Community (or glocal), and Personal (local). Where the personal

layer represents the user's local knowledge from his/her personal point of view, the Community represents knowledge shared by several users (and there can be several communities around diverse areas of interest), and the Universal represents knowledge shared across communities, and therefore by many users. We define the structure the knowledge base at all three layers to have the following three components (for more details see Chapter 3 "Knowledge Components"):

Background knowledge: containing the conceptual and linguistic parts.

Schematic knowledge: containing the etypes and attribute definitions.

Concrete knowledge: containing entities.

The remainder of this chapter is organized as follows: in Section 7.2 we define the knowledge propagation problem; in Section 7.3 we will see a motivating example of how this propagation of knowledge can happen, considering two different scenarios for the learning user; in Section 7.4 we will present the related work. Given the three main components of the knowledge (background, schematic and concrete) defined in Chapter 3, we dedicate one Section for each component in this Chapter presenting a pragmatic approach to address the propagation of the knowledge between the users. In Section 7.5 we first define how to propagate the language and the concepts; in Section 7.6 we then define how to propagate attribute definitions and entity types and in Section 7.7 we will see how to propagate entities. Finally, Section 7.8 summarizes the Chapter.

7.2 Problem definition

In order to minimize the difference between the knowledge of different users, during the initial process of bootstrapping of their knowledge bases, all users acquire "basic" Universal Knowledge (UK) that will enable interoperability. The UK can be seen as commonly shared and well understood knowledge curated by communities of experts (each of them on their own domain of expertise). By "basic" knowledge we mean an initial language and the corresponding set of concepts with common sense entity types such as Person, Organization, Event and Location. Users could also choose to bootstrap (or learn initially) only some parts of the UK in the domain of their interest. For example, when bootstrapping knowledge from the domain of medicine, users get extra linguistic information containing technical terms, for example, for "heart attack" they also get "myocardial infarction" and "cardiac arrest".

Given the common origin of the knowledge of the users (the UK), much of the knowledge is already aligned and users can safely exchange knowledge which will be understood. However, there are several cases when there can be a difference in the knowledge between users, in any of its components, which would require further actions in order to achieve interoperability. The problem can be defined as follows:

User A wants to import a new piece of knowledge (k^n) from user B, e.g., entity $E1$. $E1$ (the k^n) is defined (even partially) in terms of a delta knowledge (Δk) contained in user B with respect to user A (there is a difference in the knowledge of the users). There are two cases for Δk:

1. Δk is defined in terms of the UK.
2. Δk is defined in terms of new knowledge defined in user B.

It is clear that from the definition above, k^n being new to user A, is also part of the difference of knowledge between user A and B, i.e., $k^n \in \Delta k$. The notation highlights only the fact that k^n is something new being learned, and it can be defined in terms of something that is not directly understood, that has to be learned too.

7.3 Motivating Example

Alice (User A) is a new user of the system, and bootstraps her knowledge base with some basic knowledge base from the UK. Bob (User B) is an experienced user whose main interest is Computer Science (and who has in the past also bootstrapped his knowledge from the UK). Figure 7.1 shows two fictitious background knowledge examples for Alice and Bob, where each box represents a concept, the label inside the box represents one of the words for the synset of the concept (in English), and the links between the nodes represent relations between concepts (*part-of*, *is-a*); the dotted (...) labels represent more nodes and more labels on that branch of the knowledge. We can see from Figure 7.1 that the Δk of Bob with respect to Alice are the nodes "Object Oriented Programming Language" and "Java".

Now assume Bob meets Alice and start talking to her about this brand new thing he invented called Java (Java being the k^n), he goes on and tells Alice that Java is a Programming Language, but not any kind of programming language, is a "Object Oriented Programming Language", which is something very new and exiting in Computer Science.

At this point we draw two possible scenarios for Alice:

Scenario 1: Alice is a sociologist that has to help Bob in writing a part of a project proposal. She listens carefully to Bob but since she is not much interested in all the details, she just grasps a general understanding of what this Java thing is, only to the extent of what is needed for helping Bob write the social impact section of the project. After the exchange of information all what Alice remembers is that Java is a Programming Language, which is something related to Computer Science. The left side of Figure 7.2 show the final state of the knowledge base of Alice.

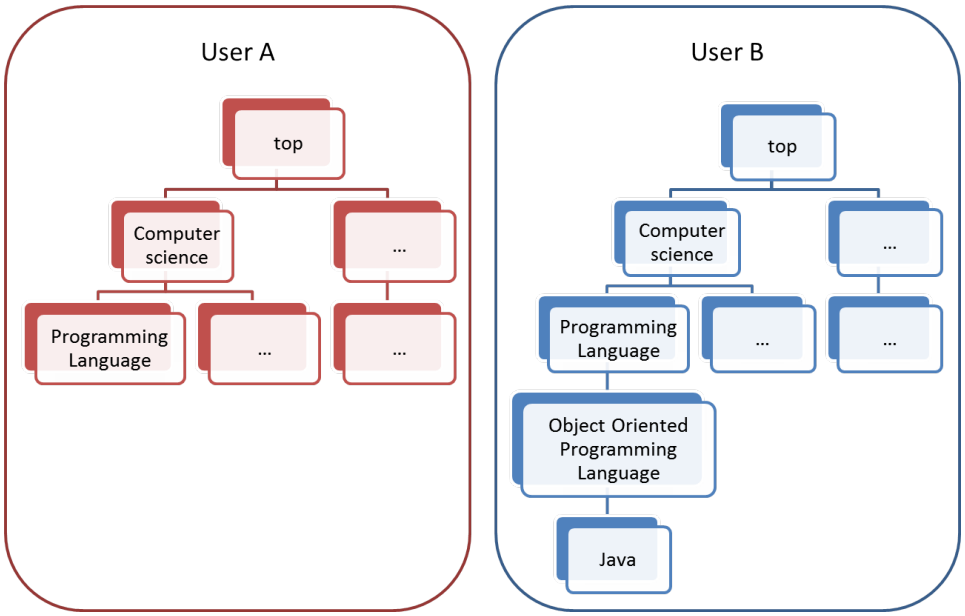


Figure 7.1: A simplified example of the knowledge of two users, A and B

Scenario 2: Alice is a first year Computer Science student, who is attending a Programming Language introductory lesson given by Bob. In this case Alice pays very close attention to all the details Bob gives about this exiting new Java Programming Language, and given that she has to pass the final exam for the course, she even learns that Java is an Object Oriented Programming Language. The right side of Figure 7.2 show the final state of the knowledge base of Alice.

From the scenarios above we can note that what Alice learns, depends on the situation. Regardless of the situation, however, is the fact that whatever she learns, has to be situated with respect to what she already knows.

In general terms there are 3 naïve approaches to address the learning problem, assuming there is a pre-shared knowledge (in our system design, the UK):

1. Drop all the Δk knowledge used to define k^n. In our example above this would mean that Alice will remember there is something called Java that Bob told her about, and no more than that (just a set of characters with no formal meaning).

2. Copy all new Δk knowledge used to define k^n. This means that scenario 2 is supported by the system, but not scenario 1.

3. Try to keep the pre-shared knowledge as much as possible. (a trade-off between 1 and 2). In this case, according to how we define the learning function, we can simulate scenario 1 and 2, namely, in some situations we need a way to propagate only some parts of the Δk in order to understand k^n.

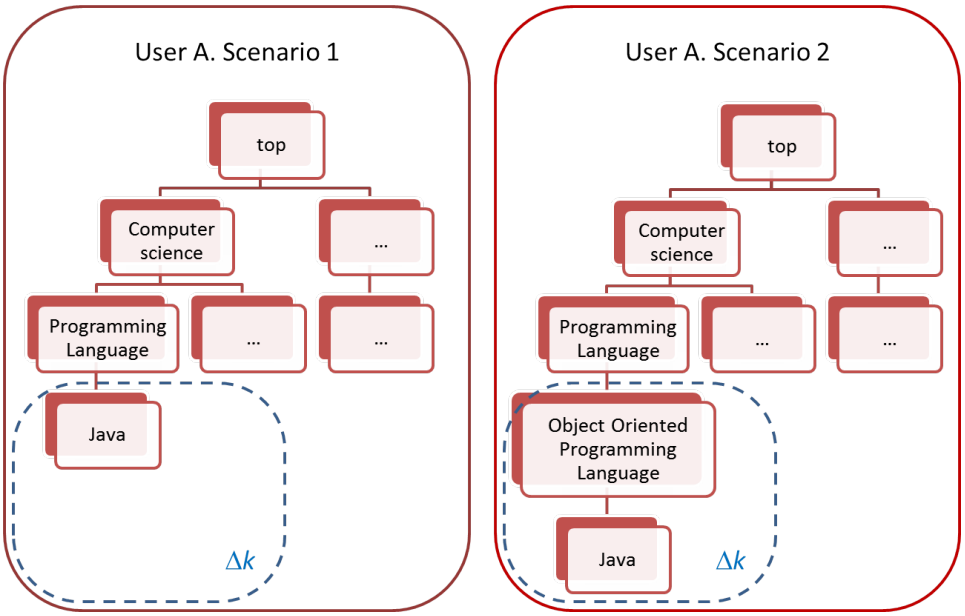


Figure 7.2: User A's (Alice) knowledge after the interaction with User B (Bob) in two different scenarios.

For pragmatic reasons we adopt the third approach. The basic idea is that, if not explicitly asked by default we only copy the minimal possible knowledge. This will prevent users from having extra knowledge that they did not explicitly asked for, while still being able to capture the formal semantics of the new knowledge being imported. In Sections 7.5, 7.6 and 7.7 we will see how this approach is implemented in each of the three components of the knowledge.

7.4 Related Word

In the field of Artificial Intelligence and Agents, the work on Meaning Coordination [SK05] is related to the problem of knowledge propagation. The problem is stated as follows:

> Two Agents A_1 and A_2 want to interoperate. Each agent has its own knowledge specified according to his own Ontology. The concepts in the agent's ontologies are considered semantically distinct a-priory from the concepts of other agents' ontologies, even if syntactically equal. Agent's ontologies are not open to other agents for inspection, therefore agents can only learn concepts from other agents through interaction [SK05].

The problem of meaning coordination is then addressed by exchanging instances of those concepts between the agents and by being able to classify the concepts in a similar manner by both agents.

As we saw in the definition of our problem in Section 7.2, we defined our knowledge propagation problem differently in the sense that there is a common pre-shared knowledge, the UK, which we can be used to explain other agents the meaning of unknown concepts. It is evident that any agent can define new concepts without any relation to the UK, but in doing so, the reasoning capabilities of the knowledge base will not be exploited, and thus, agents (in our case, human agents) would be encouraged to create new knowledge in terms of pre-shared knowledge. However, in case the agent still defines new knowledge in isolation, the state of the art approaches to meaning coordination can be applied, but this is out of the scope of the current thesis Chapter.

Finin et al., [FFMM94] present a Knowledge Query and Manipulation Language (KQML) defining the syntax and semantics to exchange information between agents. The work recognizes three fundamental component for achieving interoperability of agents: *i)* a common language; *ii)* a common understanding of the knowledge exchanged; and *iii)* the ability to exchange whatever is included in (i) and (ii). In the terms of our work the common language is defined in Chapter 3, the common understanding of the knowledge is given by the UK, and the ability to exchange knowledge is the purpose of this Chapter. We do not commit to any explicit language to exchange information between agents, however, for any such standard to be considered, it needs to comply with the characteristics of our Knowledge, outlined in Chapter 3 (see Section3.5 for a discussion of some relevant languages).

The authors in [BRIW06] analyze the economics of knowledge exchange, i.e., what is the process by which knowledge starts as a single person idea and spread to become shared among individuals. They presents a proof that knowledge exchange is most effective when agents are alike (i.e., they share some knowledge) and thus interaction is made easier. However, it states that if knowledge between users is too diverse, there is no room for understanding, but if knowledge is too similar, now real new knowledge (or ideas) can be created.

Finally, we need to draw a difference in the literature of agents and the exchange of knowledge as defined in Section 7.2. Much work is devoted to learn the best behaviour (the actions that need to be taken by an agent, see for example [Bon01]). While normally referred to as agent learning, the object of what is being learned in these works is different from the current Chapter, in most cases, they refer to adaptation and not knowledge propagation as we defined earlier in the Chapter. While the philosophy is the same, the need to achieve understanding and coordination between agents (see for example [BRIW06]), the pragmatics are different.

7.5 Background Knowledge

We take a pragmatic approach to the propagation of knowledge given our definition in Section 7.2. Considering the three components of the knowledge defined in

Chapter 3, we can see that the most basic component is the Background Knowledge (BK). The schematic knowledge is then defined in terms of the BK, and the concrete knowledge uses both the schematic and background knowledge. In this and the following two sections, we define how to treat the propagation of knowledge starting from the BK, then followed by the schematic and the concrete knowledge.

The Background Knowledge (BK) component consist of the conceptual and linguistic parts. The conceptual part is composed by concepts and relation between concepts, and the linguistic part is composed of several possible languages, each of them composed by words grouped in synsets, which are related to concepts; where a word can be part of several synsets (polisemy). See Chapter 3 Section 3.2 for details.

Therefore, in the case of the background knowledge, the possible new knowledge (k^n) to be propagated are concepts (and its relations) and linguistic elements (words and synsets), and the Δk knowledge is also defined in terms of concepts and linguistic elements (a new concept could have been defined in terms of another new concept). For example, in the right side of Figure 7.1 we can see that Bob define a new concept for "Java" (the k^n) which is defined in terms of another new concept, "Object Oriented Programming Language" (the Δk, with respect to Alice).

Given the one-to-one relation between concepts and synsets[1], the basic elements we need to define how to propagate are new concepts (with its synset) and new words. We first define a basic function that is used to "explain" new concepts and then show how to use this function in order to learn concepts and words.

7.5.1 Explaining new concepts

When first instantiating his/her knowledge base, the user imports knowledge from the UK. The user could also choose to import only relevant domains such as Medicine, Music, Sports, etc. This means that in general, all users have (a portion of) the same basic conceptual and linguistic parts.

When a new concept is not understood by a user, say user A, s/he can ask the other user, say user B, to explain this new concept to him/her. The basic idea is to enrich the user A's knowledge with the minimal concepts and relations which allows to give meaning to the new words and concepts during the exchange of information with user B.

In order to implement this idea, we defined the following function:

Compute path of a given concept to a UK defined concept

$$compute\_path\_to\_uk(C_N) \rightarrow path \tag{7.1}$$

Computes the shortest path between:

[1] a new concept implies a new synset, otherwise people cannot talk about the concept, and a new synset implies a new concept, otherwise if people just want to call an existing concept in another way, they just need to add a new word to the existing synset, assuming the same language.

- a new concept C_N (the k^n at BK level) defined by user B and unknown by user A (the one that imports),
- and a concept defined in the UK that is already in user B's conceptual part, let us call it C_{UK}.

The function returns a $path$ (as compute by user B) where the nodes are concepts and the links are the relations between the concepts. The starting point of the path is C_N and the end is C_{UK}, and there can be zero or more other new concepts $\{C'_N\}$ in the path defined by user B in order to explain C_N to user A. For each unknown concept by user A (C_N and $\{C_{N'}\}$), the result also includes the synset and its corresponding words.

In the example of Figure 7.1, if Alice asks Bob to explain "Java" to her, the result of this function would be the path composed by $\{$*Java* $\rightarrow$ *Object Oriented Programming Language* $\rightarrow$ *Programming Language* $\}$.

If C_N has not link to any concept defined in the UK, then the path contains C_N as result.

7.5.2 Basic learning operations

There are two basic cases for leaning BK's components, these are: new concepts and new words for existing synsets. Considering the words, we focus only on semantic strings and word senses. Words that have no semantics attached are considered as any other basic datatypes (integers, dates . . .), in this case, a set of characters.

7.5.2.1 New concept

Defined in terms of the UK In this case user A asks user B to explain the unknown concept calling the $compute\_path\_to\_uk$ function defined by equation 7.1. Then the missing concepts, relations, synsets and words obtained from the method are added to user A's background knowledge.

It can be the case that the end concept C_{UK} returned by the function is not known by user A (because it is defined in a domain that was not imported by user A). In this case user A calls again the $compute\_path\_to\_uk$ with the previous C_{UK} as parameter until a known concept is returned.

Not related to existing concepts from the UK Since user A cannot know that the new concept is not related at all to other existing concepts form the UK, user A has to call the $compute\_path\_to\_uk$ as in the previous cases. The result will only contain the unknown concept and the corresponding synset and words, which can be added to the local background knowledge.

7.5.2.2 New word for synset

Extending existing synsets for existing concepts (from the UK) In this case we add the new word to the local background knowledge.

For new concepts (implies Case 7.5.2.1) In this case we need to perform the same operations as in the concept case, see Section 7.5.2.1

7.6 Schematic knowledge

The schematic knowledge is composed by the entity type and attribute definitions. These components are in turn defined in terms of the background knowledge elements (concepts and words, see Chapter 3 Section 3.3 for more details). As we will see in the following subsections, given that we know already how to propagate BK elements, we can now define how to propagate new attribute definitions and entity types definitions.

7.6.1 New attribute definitions

An attribute definition AD, defined in Model Object 9 in page 20 contains a concept c, the domain D of the values and a definition of whether it allows one or multiple values. Considering this definition, we need to recursively propagate also the concept c, if missing. The following are the possible cases when propagating an attribute definition:

Based on existing concepts In this case we can safely copy the new attribtue definition given that we already know the concept.

Based on new concepts In this case we first apply the solution for copying the new concept (Case 7.5.2.1) and then we apply the solution for copying new attribute definitions for known concepts.

7.6.2 New entity types

An entity type $ET = \langle c, \{AD\}\rangle$ as defined in Model Object 8 in page 20 contains a concept c and a set of attribute definitions $\{AD\}$. The following are the possible cases when propagating an entity type definition.

Based on existing concepts Given that we know already the concept being used to define the entity type, we can copy the new entity type definition without problems.

Based on new concept We first copy the new concept as defined previously in Section 7.5.2.1. Once we already learned the missing concept, we can apply the same solution for copying new entity types based on existing concepts.

Based on existing attributes Similarly to the case of existing concept, we can copy the entity type reusing the attribute definitions.

Based on new attributes In this case we first apply the solution for copying the new attribute definition. Once we already have the attribute definitions, we can copy the entity type as in the previous cases.

Note that more than one combination might be required when copying a new entity type, for example, it might be defined based on a new concept and a new set attribute definitions. The solutions presented are not exclusive among each other.

7.7 Concrete knowledge

The concrete knowledge is what users are most likely to define, given our vision of a semantic system. Users would want to create entities for their contacts (people), events, locations and businesses they like, and then share these entities with others.

An entity, is defined by Model Object 10 in page 26 as $e = \langle ID, SURL, etype, name, EM, [link], [copy] \rangle$. The propagation of the elements related to the identity of the entity $\langle ID, SURL, name, [link], [copy] \rangle$ is defined in Chapter 9 "Identity Management". Also, the propagation of the entity itself, i.e., the act of coying the entity, is defined in Section 9.3.2. Here we focus on the propagation of the entity metadata EM element.

As defined by Model Object 11 in page 27, the entity metadata ($EM =< \{Attr\} >$) is composed by a set of instantiated attributes. The instantiated attribute is defined by Model Object 12 in page 27 as ($attr = \langle AD, AV, T \rangle$), composed by an attribute definition AD, an attribute value AV and a time dependency T.

The propagation of new attribute definitions was already discussed in Section 7.6, here we will define the propagation of the missing part, the attribute values. We distinguish between semantic, semantic-less, and relational attribute values.

7.7.1 New Attribute values

Semantic-less attribute values A semantic-less attribute value is an attribute value defined in the domain of a basic datatype such as String, Integer, Float, DateTime. Since these values are no related to the concepts in the conceptual part or any other parts of the knowledge, we can copy the values with no further implications.

Semantic attribute values By semantic attribute value we mean a word with a link to a concept in the conceptual part of the BK (a word sense as defined in Model

Object 4 in page 17) or a semantic string (as defined in Model Object 5 in page 17). If the value is a known concept and the word is already in the corresponding synset, we copy the attribute value without further issues. If the value is an unknown concept or the word is not in the corresponding synset, we apply first the solution for Case 7.5.2.1 (New concept), and then we copy the attribute value.

Relational attribute value based on existing entities In this case the value is a reference to an entity which is already known. This implies that the ID of the referred entity is a $SURI$, for which we already have a local set of metadata describing it, identified by local $SURL$. In this case we copy the attribute value for the entity being copied, but we change the local value in user A to point to the $SURL$ of the local existing entity.

Relational attribute value based on unknown entities In this case the value is a reference to an entity which is unknown by user A, i.e., the ID of the entity being referred is a $SURI$ for which there is no local $SURL$ with no local set of metadata.

In order to keep in control the transitivity and the local knowledge of user A, we do not copy this relational attribute value. This means that when copying entities, we do not transitively copy the unknown entities.

7.8 Summary

In this Chapter we have proposed a methodology for propagating knowledge between users. We assume a scenario were all the interacting users have the same knowledge structure, and this structure is given by the three components defined earlier in Chapter 3 "Knowledge Components". In this scenario, when users bootstrap their knowledge, they can reuse the knowledge at the Universal layer (UK) and from there start defining their own local knowledge to fit their application domain or needs.

With the above mentioned assumptions, the difference in the knowledge between users can be explained in terms of the knowledge from the UK. This is a key feature of the architecture defined in this thesis, making the problem of knowledge propagations to be grounded in the capability of users of explaining each other concepts in terms they can understand.

We envision that most of the propagation will be activated when copying entities between users (for example: contacts, events, locations). Assuming this as an entry point, our pragmatic, automatic, approach is then based on the following actions:

1. we copy the new entity,

2. we copy attributes values,

3. for relational attribute values, we copy known entities,

4. we enrich the knowledge with the minimal information that allows to give meaning to the terms used in the entity type, attribute definitions and the semantic values.

Is it clear that what will be actually learn, once the explanations are given, is dependent on the situation where the learning is taking place. But given the general procedure for explaining defined in this Chapter, the learning user can adapt the learning part to fit the situation and application domain.

Part IV

Entity compatibility

Chapter 8

Entity Compatibility

8.1 Introduction

In Chapter 3 Section 3.4.3 we have presented an approach of solving the issues related to using names as references for real world entities in the mental and digital entities, recognizing the need of local identifiers, that we call $SURL$, and global identifiers for the real world entity, that we call $SURI$.

In this section we present our approach to solving the locality issue of the definite descriptions in mental models (which leads to *incomplete* and *inconsistent* knowledge about real world entities). We tackle this issue in order to be able to check whether based on their metadata EM two entities refer to the same real world entity or not, and assign or reuse a $SURI$ correspondingly.

The main issue with definite descriptions in mental entities is that the uniqueness is with respect to all the local entities of the bearer of the mental entities; therefore, is the creator of the description which decides more or less arbitrarily the criteria that identifies one entity from the others (which also applies to names) [Sea58]. Thus, given two mental models about the same real world entity[1], if they differ significantly from one another, the interaction between people holding these mental models will be difficult [Ful94][2]. In order to avoid this we propose a set of compatibility rules based on identifiers ($SURL$, $SURI$ and natural languane name) and metadata, with 3 different possible outcomes:

1. Compatible: when we can be sure two entities refer to the same real world entity.

2. Not compatible: when we can be sure two entities refer to different real world entities.

3. Undefined: when we cannot be sure as to whether the two entities being compared refer to the same real world entity.

[1] from different people

[2] They will not know they are referring to the same real world entity

The aim of this Chapter is to define the conditions under which we can consider that two digital entities refer to the same real world entity. We do this based on the available references (names and identifiers) and the the description encoded in the metadata EM. To achieve this, we require an algorithm with near perfect precision, and where recall is not as important as precision. We need this high precision because the algorithm is to be used to know when to assign a new globally unique identifier that refers to a real world entity (the $SURI$), or to reuse one already existing for the same real world entity, perhaps using a slightly different description.

The novelties of our approach are *i)* the recognition of the importance of the name as a reference and *ii)* the identification of a set of descriptive attributes that can uniquely describe *globally* the real world entity being referred. By separating the name from the other descriptive attributes we can apply advance name matching techniques to compare them, and by identifying (sub)sets of available attributes as globally identifying we can reduce the comparison time and test several combinations, as opposed to use all the available attribute values at the same time to compare the digital entities.

The rest of the Chapter is structured as follows. In Section 8.2 we outline the compatibility principle for identifiers and descriptions. In Section 8.3 we compare our approach to the other related approaches. Finally, section 8.4 summarizes the Chapter.

8.2 Principle of compatibility

In this Section we define the conditions under which two digital entities are considered to refer to the same real world entity. We first study the compatibility on identifiers, i.e., references to the real world entity. In this category names are of special interest as they are treated separately from descriptions. By doing so, we can apply state-of-the-art name matching techniques to compare them. We later study the compatibility on descriptions. We defined the notion of identifying set that allows us to overcome the issue of local identification of definite descriptions.

8.2.1 Compatibility on identifiers

The compatibility on identifiers refers to the compatibility of elements of the digital entity that act as references to the real world entity, i.e, names, *SURL*s and *SURI*s. Given that comparing digital identifiers ($SURL$ and $SURI$) is a mater of exact string (or numerical) match, we first spend some words about the issue of comparing names, and later, we will use these notions to define the compatibility principles on identifiers.

8.2.1.1 Comparing names

As we have seen, the role of the name is to refer to a real world entity. People use several types of names, the most common being proper names. When comparing names there are several types of variations that need to be addressed [Chr06]:

1. **Language variations**: Proper names of entities (normally famous entities) tent to be translated. For example: *Napoleon* vs. *Napoleone*, *London* vs. *Londra* vs. *Londres*, *Pope John Paul II* vs. *Papa Juan Pablo II* vs. *Papa Giovanni Paolo II*. As can be noted in the example, a basic exact or approximate string comparison technique will not suffice to detect that two versions of proper names for the same real world entity in different languages are the same.

2. **Syntactic variations**: There is a wide range of possible syntactic variations that can occur when names are written. To cite a few we can find capitalization (*Ibm* vs. *IBM*), punctuations (*Mc'Donald* vs. *Mc Donald*), abbreviations (*HP* vs. *Hewlett Packard*) and misspellings.

3. **Format Variations**: This category is normally present in personal names where we can find several formats such as:

   - first_name last_name
   - first_name middle_name last_name
   - first_name abbreviation_of(middle_name) last_name
   - abbreviation_of(first_name) abbreviation_of(middle_name) last_name
   - last_name, first_name
   - last_name first_name

   The Wikipedia article for Personal Names[3] contains a rather extensive list of how personal names change according to cultures.

4. **Phonetic variations**: such as "Sinclair" vs. "St. Clair".

5. **Alternative names**: such as "Alexander III of Macedon" vs "Alexander the Great", Pope John Paul II vs. Karol Jósef Wojtyła. Nicknames also fall under this category, but we here consider only those that are widely used, as the example shows.

From the list above, we can see that comparing proper names is not an easy task. Several research initiatives report different techniques [Bra03, Chr06] and book has been dedicated to the issue [HD04].

We adopt in initial pragmatic approach in which we deal with the translations and alternative names using a dictionary-like approach. Depending on the

[3] `http://en.wikipedia.org/wiki/Personal_name`

entity type we can get the variations from sources like the Internet Movie Database (IMDB)[4] for movies, actors and directors, or from Geonames[5] for locations.

The other variations are deal with using an approximate matcher. Given the evaluation of possible name matchers in [Chr06], the best string comparator is PermWink. The reader is referred to the paper to a complete account of the comparison algorithm.

Given the difficulty of the name matching task, we leave as future work a complete account on the issue of how to best treat the name matching problem, also considering performance issues, as the matching should normally be done on-line. The name matching problem on its own represents a whole research area out of the scope of the current thesis.

Having defined our initial approach for how to compare names, in the following sections we define the states of the principle of compatibility applied to identifiers.

8.2.1.2 Compatible

There is compatibility on identifiers in the following cases (in the specified order):

1. the $SURL$ is the same (no need for compatibility on metadata)
2. the $SURL$ is different, but the IDs is the same ($SURI$ or $SURL$), (no need for compatibility on metadata).
3. the name is the same (or similar[6]), and there is compatibility on metadata.

8.2.1.3 Non-compatible

There is non-compatibility on identifiers in the following cases (in the specified order):

1. the $SURL$ is different, and
2. the IDs is different, and there is non-compatibility on metadata.
3. the name is significantly different, and there is non-compatibility on metadata.

8.2.1.4 Undefined

There is undefined compatibility on identifiers in the following cases (in the specified order):

1. the $SURL$ is different, and

[4] `http://www.imdb.com/`

[5] `http://www.geonames.org/`

[6] Where the threshold has to be set experimentally and is left as future work.

2. the IDs is different, and the ID is a $SURL$, and there is undefined compatibility on metadata.

3. the name is significantly different, and there is undefined compatibility on metadata.

8.2.2 Compatibility on descriptions

Definite descriptions (coming from mental entities) have a unique referent with respect to all the other mental entities of the same person, i.e., their identifying power is only local. However, in order to capture the uniques property of real world entities, and translate this property into the uniques of our digital counterpart of the real world entity identifier, the $SURI$, we need a way to globally distinguish entities from one another. The compatibility criteria presented in the following subsection deals with this issue.

8.2.2.1 Identity criteria

The digital counterpart of a definite description is what we call **identifying sets** of attributes. An identifying set is a set of attribute definitions $\{AD\}$ defined for each entity type. When the identifying set is instantiated with the attribute values of an entity, it allows us to globally distinguish entities (of the same type) from one another. For example, for entities of type "Person" the values of the attributes "passport number and the issuer" uniquely identify people from one another, and therefore it can be thought as an identifying set for the "Person" entity type.

In what follows we give a more formal definition for the identifying set and some simple algorithm to compute them, given the availability of an entity dataset.

> **Model Object 13** (Identifying Set ($iSet$)) An identifying set is a tuple $iSet = \langle id, etype, \{AD\}, conf \rangle$ where id is the identifier of the identifying set, $etype$ is the entity type the identifying set applies to, $\{AD\}$ is a list of attribute definitions that are part of the set, which should be a subset of the attribute definitions defined in the referenced $etype$, and $conf$ is the degree of confidence or certainty with which the set uniquely identifies an entity, as opposed to any other possible combination of attributes.
>
> The purpose of the identifying set is to group a minimal set of attributes that uniquely and globally "identifies" an entity from all the other entities of the same type. There might be several $iSet$s for each entity type, and attributes can be shared among $iSet$s. For example, a person might have several $iSet$s such as $\langle 1, Person, \{name, email\,address\}, 100 \rangle$ or $\langle 2, Person, \{name, mobile\,phone\,number\}, 100 \rangle$.
>
> We say that an entity e is **well identified** (or that the entity is identifiable) if there is at least one subset of instantiated attributes $\{Attr\}$ in EM whose attribute definitions are part of an identifying set for its particular $etype$.

Conversely, an entity e is **not well identified** (or the entity is not identifiable) if there is no subset of instantiated attributes that can comply with an identifying set for its $etype$.

The exact definition of the identifying set is normally domain dependent, as it relies on the existing attribute definitions on the entity type (which is domain dependent as we previously discussed in Section 3.4.3). One could normally define manually several identifying sets of attributes based on experience and common sense. However, setting the value for the confidence measure can be difficult if done manually. A suggested method for defining the confidence for each identifying set in each entity type could be the one presented in Algorithm 1. The algorithm assumes there existence of a large corpus of entities for the specified etype (example of such corpus are YAGO[7] or GeoNames).

Algorithm 1 Simple (greedy) algorithm to compute the confidence of an already defined identifying set)

Require: $e \in E$ {a corpus of entities E for a particular etype}
Require: $iSet = \{attr\}$ {an identifying set which is composed as a set of attributes}
Require: $get\_attr\_val(e, iSet) = \{attr\_val\} = identifying\_values$ {a function get_attr_val that given an entity e and an identifying set $iSet$ returns a set of attribute values $\{attr\_val\}$ for those attributes in $iSet$ in e }
Require: $count\_entities(identifying\_values, \{e\})$ {a function $count\_entities$ that given an a set of attribute values $identifying\_values$ counts how many entities in $\{e\}$ contain the same set of attribute values}
1: $M = \emptyset$ {A set for containing the counts of matching entities}
2: **for all** $e_j \in E$ **do**
3: $identifying\_values_j = get\_attr\_val(e_j, iSet)$
4: $match_j = count\_entities(identifying\_values_j, E - e_j)$
5: $M = M \cup match_j$
6: **end for**
7: $avg = average(M)$
8: $conf_{iSet} = 1 - avg/|E|$

If there is no identifying set already defined for a specific etype, still assuming there is a corpus of entities for the particular etype, one could create a simple greedy algorithm that tests all the combinations (without repetition) of available attributes (considering those attributes that have values in at least X% of the entities, e.g., 50%), let us call a particular combination c. Algorithm 2 shows this methodology where the confidence of each combination of attributes is computed, and if this confidence is greater than a given threshold (which can be considered the error allowed in the application domain), then the combination is considered to be an identifying set. Line 11 of the algorithm tries to optimize the procedure by breaking the computation of how many other similar entities there are with the same values when there is enough evidence that the given combination is not identifying enough.

[7] `http://www.mpi-inf.mpg.de/yago-naga/yago/`

Algorithm 2 Simple (greedy) algorithm for computing the identifying sets for a particular etype.

Require: $e \in E$ {a corpus of entities E for a particular etype}
Require: $at^{Et} = \{attr\}$ {a set of attributes avalilable for a specific etype Et}
Require: $get\_attr\_val(e, at^{Et}) = \{attr\_val\} = identifying\_values$ {a function get_attr_val that given an entity e and a set of attributes at^{Et} returns a set of attribute values $\{attr\_val\}$ for those attributes in at^{Et} in e }
Require: $count\_entities(identifying\_values, \{e\})$ {a function $count\_entities$ that given a set of attribute values $identifying\_values$ counts how many entities in $\{e\}$ contain the same set of attribute values}
Require: $combination(at^{Et})$ {a function $combibation$ that returns all the combinations (without repetition) from a set of attributes at^{E}}
Require: $threshold_w$ {a threshold for deciding when a set that contains a particular weight could be considered as an identifying set, e.g., 0.8}
$I = \emptyset$ {the set to contain the identifying sets}
$check = |E| * 0.1$ {this parameter serves for checking every 10 percent of the total number of entities whether it is worth to continue checking the other entities for duplicates}
for all $c_k \in combination(at^{E})$ **do**
  $M = \emptyset$ {A set for containing the count of matching entities}
  for all $e_j \in E$ **do**
    $identifying\_values_j = get\_attr\_val(e_j, c_k)$
    $match_j = count\_entities(identifying\_values_j, E - e_j)$
    $M = M \cup match_j$
    if $|M| \% check = 0$ **then**
      if $average(M)/|E| > ((1 - threshold_w) * 2)$ **then**
        break the for {if at a certain point the average of similar entities for the particular combination or attribute is already twice the allowed error, then this combination is considered not identifying enough}
      end if
    end if
  end for
  $avg = average(M)$
  $confidence_{c_k} = 1 - avg/|E|$
  if $weight_{c_k} > threshold_w$ **then**
    $iSet = \langle k, etype, c_k, weight_{c_k} \rangle$
    $I = I \cup iSet$
  end if
end for

Given our notion of identifying sets, the **Compatibility** principle on descriptions is defined in the following subsections using a recursive approach based on the structure of entities.

8.2.2.2 Compatible

There is compatibility on descriptions (metadata) in the following cases (in the specified order):

1. the $etype$s are compatible, and metadata EM are compatible;
2. two $etype$s are compatible if their concept match;
3. two EMs are compatible if there is at least one subset of $\{Attr\}$ that forms an identifying set $iSet$ that is compatible, and there is not subset that forms an identifying set that is non-compatible. The remaining attributes that are not part of identifying sets do not affect the compatibility.;
4. two $iSet$s are compatible if all of the contained instantiated attributes $\{Attr\} \in iSet$ are compatible;
5. two instantiated attributes $Attr$s are compatible if their attribute definition AD and the attribute value(s) AV are compatible;
6. two ADs are compatible if their concept C match, and the domain D are comparable. E.g., integers are comparable with float, but not with "semantic strings".
7. two AVs are compatible if the values are compatible using the basic data type comparator (dates, integers, concepts, coordinates, URLs) or the values are within the margin of error allowed by the $etype$ definition (e.g., for the Person $etype$, birth dates are within the same day and the values of height are the same up to centimeters, for Location the latitude and longitude are the same up to the 4th decimal, etc.)[8].

8.2.2.3 Non-Compatible

There is non-compatibility on metadata in the following cases (in the specified order):

1. the $etype$s are non-compatible, or the metadata EM are non-compatible;
2. two $etype$s are non-compatible if their concept do not match;
3. two EMs are non-compatible if there is at least one subset of $\{Attr\}$ that forms an identifying set $iSet$ that is non-compatible;

[8]The complete list of "fuzzy" comparators is to be defined by each $etype$ developer as it is normally domain dependent

4. two $iSet$s are non-compatible if at least one of the contained instantiated attributes $\{Attr\} \in iSet$ is non-compatible;

5. two instantiated attributes $Attr$s are non-compatible if their attribute definition AD are compatible and the attribute value(s) AV are non-compatible;

6. two AD are compatible if their concept C match, and the domain D are comparable.

7. two AV are non-compatible if $iSet$ in AD is $false$ and if the values are non-compatible using the basic data type comparator (dates, integers, concepts, coordinates, URLs) or the values are not within the margin of error allowed by the $etype$ definition.

8.2.2.4 Undefined

There are several cases that falls in the middle of the above compatibility and non-compatibility definitions, for which we say that we **don't know** whether the entities are compatible or not. In these cases user intervention might be required.

There is undefined compatibility on metadata in the following cases (in the specified order):

1. the $etype$s are compatible, and there is undefined compatibility on metadata EM;

2. two $etype$s are compatible if their concept match;

3. two EMs have undefined compatibility if the set of attribute definitions $\{AD\}$ is disjoint, or there is no common subset of $\{Attr\} \in EM$ to form a compatible $iSet$, or if the identifying sets $iSet$ have undefined compatibility;

4. two $iSet$s are have undefined compatibility if at least one of the contained instantiated attributes $\{Attr\} \in iSet$ has undefined compatibility;

5. two instantiated attributes $Attr$s have undefined compatibility if their attribute definition AD have undefined compatibility, or AD is compatible and the attribute value(s) AD have undefined compatibility;

6. two AD have undefined compatibility if their concept c match and the domains D are non-comparable (E.g., one D is defined as a string and the other as float).

7. two AV have undefined compatibility if $iSet$ in AD is $true$ and if the values are non-compatible using the basic data type comparator (dates, integers, concepts, coordinates, URLs) or the values are not within the margin of error allowed by the entity type definition[9].

[9] In this case we are in the presence of partial information, as it could be compatible in other values, e.g., phone numbers.

8.3 Related Work

The issue of recognizing whether two different descriptions of entities refer to the same real world entity is known with many names in different research areas. In databases is referred to as record linkage, entity consolidation and duplicate detection, to name a few [HZU+12]. Typically, these approaches focus on string similarity and statistical analysis to detect duplicate references to the same real world entity. Given its statistics roots, there is room for errors, while in our approach, we aim at maximizing precision first, i.e., we need to be certain that the metadata of different entities refer to the same real world entity in order to reuse or assign a new real world entity identifier (a SURI) to the given digital entity.

Much work has also been done in the Semantic Web Community commonly referred to as entity matching [HZU+12] or instance matching [EFM+10]. A widespread approach in this community is to rely on the owl:sameAs relation [ABK+08, HZU+12], functional and/or inverse functional properties[10] [HHD07, HZU+12]. However, the semantics of owl:sameAs predicate defines that the URIs linked with it have the same identity [DS04], i.e., must be exactly the same resource with respect to all properties (the same perspective). The major drawback of this approach is that the two URIs become indistinguishable even though they may refer to different perspectives of the same entity according to the context in which they are used [JGM08]. This also implies that there is no clear separation between the identifier of a set of properties (or description) of an entity, and the URI of the real world entity itself, as recommended in the W3C Note "Cool URIs for the Semantic Web" [SC08].

Furthermore, usually the goal of semantic web tools working with Linked Data[11] are oriented towards query answering. In these cases precision is not at the top of the requirement of the systems, relying on the end user the task of discarding manually incorrect results (as we can see in Sindice[12]) or by blaming the data for the incorrect results stating that "incorrect consolidation is due to errors in the data, not an erroneous approach" [HZU+12]. While this might be acceptable by their problem definition, given that we study entity compatibility for the assignment and reuse of globally unique real world entity identifier, low precision is not acceptable to our needs.

The Silk Link Discovery Framework [IJB10] relies on user-defined matching heuristics to find synonymy in URIs from different data sources referring to the same real world object. This is similar to defining manually the identifying sets for entity types, but the lack of support from the other components of the system (background, schematic and concrete knowledge) makes the implementation dif-

[10] For example, the foaf:mbox property representing the email address of a person in FOAF (`http://xmlns.com/foaf/spec/`) is defined to be unique to a person, therefore, if two instances share the same foaf:mbox property, then it could be inferred that both refer to the same real world entity.

[11] http://linkeddata.org/

[12] As the example provided by [HZU+12] shows: `http://sig.ma/search?q=paris`

ferent.

Stoermer et al., [SR09] define a feature based entity matching algorithm that recognizes the fact that certain properties are more important than others when used for identifying an entity. Although the framework claims to be entity type dependent and does not enforce any fixed schema, in their evaluations in the OAEI 2009 campaign they only matched entities of the same type with each other, with mixed performance. The work is in the line with our approach in the sense that we also recognized that only a certain set of attributes are relevant when comparing entities, the ones in the identifying set, but we use them in sets, as opposed to all the relevant ones at the same time.

One of the shortcomings discussed in [SR09] is precisely the problem of matching names. For example, they reported how normal string comparison methods such as Levenstein distance is not suited to deal with the format variations of names reported in section 8.2.1.1. As stated before, this is a proof that the name matching problem is a difficult issue *per-se*, and as such should be dealt separately.

The name matching issue has been studied in [Bra03, Chr06] and a book has been dedicated to the issue [HD04]. Several techniques are commonly used such as those based on phonetic encoding (soundex, phonex, ...), based on pattern matching (Levenshtein distance, NGram, Longest Common sub-string, Jaro, Winkler, PermWink, ...) and their combinations. According to [Chr06] the Jaro and PermWink techiniques are the best performing with an average f-measure of 0.601 and 0.883 respectively.

8.4 Summary

In this Chapter we presented our approach for addressing the issues with multiples descriptions and names from mental entities to refer uniquely to real world entities. We defined the conditions under which we say that two digital entities are compatible (i.e., we are sure the both refer to the same real world entity), incompatible (i.e., we are sure that they bot refer to different real world entities) or the compatibility is undefined (i.e., we cannot be sure whether they both refer to the same or different real world entities).

The novelties of our approach are *i)* the recognition of the importance of the name as a reference, separating it from the descriptions allowing to apply tailored name matching techniques, and *ii)* the identification of a subset of descriptive attributes that can uniquely describe *globally* the real world entity being referred, that we call an **identifying set**. By separating the name from the other descriptive attributes we can apply advance name matching techniques that exploit the characteristics of names to compare them, and by identifying (sub)sets of available attributes as globally identifying we can reduce the comparison time and test several combinations, as opposed to use all the available attribute values at the same time to compare the digital entities.

The compatibility condition differs from other entity matching and record link-

age techniques because the main purpose of the compatibility conditions is to be used by the "Identity Management" algorithm to be presented in Chapter 9 to know when to assign or reuse new globally unique identifiers for the real world entity being described by the digital entity. As such, the main requirement is precision, while recall is not as important given that the comparison method should be designed to minimize false negatives (say that the digital entity refers to different real world entities, while they actually refer to the same real world entity), while strictly avoiding false positives (say that the digital entity refers to the same real world entity, while they actually refer to different real world entities).

Chapter 9

Identity Management

9.1 Introduction

Large amounts of data about entities are generated and managed by several organizations, institutions and users on the web, dealing with different but also overlapping application domains. This information is currently publicly available thanks to the widespread adoption of Internet and application developers could take great advantage from reusing the already existing public information.

However, the diversity in knowledge, be it at the vocabulary, structure, content, and even at the identifier level, makes interoperability between applications a difficult task. In Chapter 5 (Semantic Matching) and Chapter 8 (Entity Compatibility) we have already focused on the issue of interoperability at vocabulary, structure and content levels. In this Chapter, Identity Management, we propose a methodology for dealing with the identity of perceived entities based on the compatibility criteria presented in Chapter 8: "8". We define what are the conditions under which we can generate identifiers for entities, and which kind of identifiers are needed in order to properly capture the semantics of real world entities.

The purpose of this chapter is to define a model that captures the semantics of the real world entities, but given that real world entities can only be captured and modeled by people, each with his/her own local view or perspective, we also need to model and understand the semantics of (local) mental entities. Digital entities are the way we capture real semantics by allowing the computer to bridge between the mental entities people provide about the real world entities[1].

The novelty of our approach lies in the ability to capture the semantics of both local perspectives and the uniqueness and globality of the real world entity into a single digital entity. By doing so, we can properly distinguish both levels, and assign proper identifiers accordingly. This allows us to overcome the problem of identifying local perspectives without claiming globality on the identifier [JGM08], while being able to maintain a unique global identifier from where we could obtain

[1] For a complete account on the difference between real world entities and mental entities please refer to Section 3.4.

all the possible local descriptions [SC08].

The remaining sections are organized as follows. Section 9.2 defines the identifiers at both local and global level, stating their requirements. In Section 9.3 we describe how each operation of the lifecycle of the entity (create, update and delete) affects the identity of the entity. Section 9.4 defines the procedure for merging identifiers in case we realize that two entities that were thought to refer to different real world entities, actually refer to the same real world entity. Section 9.5 presents the related work for managing identifiers on the Web. Finally, Section 9.6 summarizes the chapter.

9.2 Digital identifiers

As defined in RFC 3986 [BLFM05], the comparison methods for assigning URIs based on metadata should be designed to minimize false negatives while strictly avoiding false positives (see Section 3.4.3), i.e., we should define that two (digital) entities have the same URI if and only if we are sure they refer the same real wold entity.

As we have seen in the previous section, names (coming from mental entities as defined in equation 3.2) as identifiers present several issues. In the WWW however, several standards for digital identifiers have been created such as URIs[2], URNs[3] and URLs[4] (see Section 9.5). URIs and URNs are simple and extensible means for identifying an abstract or physical resource [BLFM05], while the URL in addition to this, provides a means of locating and dereferencing the resource. Important properties of these WWW identifiers are:

1. Global uniqueness,
2. Persistence (even when the resource ceases to exist),
3. Uniformity.

As outlined before, the aim of the digital entity, and therefore its digital identifier, is to bridge between the real world entities and mental entities (between the global and local models). This means that our framework needs to encode these two digital identifiers in order to capture the characteristics of both mental and real world entities.

9.2.1 Local and Global identifiers

While mental entities (see equation 3.2) and its name element as local identifier have the problem of not being globally unique nor being persistent, WWW iden-

[2](RFC 1630:Universal Resource Identifiers [BL94] and RFC 1738: Uniform Resource Identifier [BLFM05])

[3](RFC 2611: Uniform Resource Names [DvGIF99])

[4](RFC 1738: Uniform Resource Locator [BLMM94]

tifiers, as previously presented, also have some drawbacks that prevent them from being used as global identifiers for real world entities, namely:

- URLs provide means to retrieve **only** *local* descriptions of real world entities, but do not uniquely identify them universally [SC08].
- URIs and URNs provide universal identification of real world entities, but do not provide a description of the entity, mental nor real world model (meta-data) or name [SC08][5].
- The issue with URL as URI for identifying real world entities (such as the approach followed by OKKAM) is that they merge the identification and the description of the object (the real world entity), but one cannot force its own description or mental model to other people or users. In order to properly solve the duality between local models and global identifiers, i.e., the identification and dereferenceability issues, the framework should allow a real world entity to be described by diverse view points, without enforcing any of them.

9.2.1.1 SURL: Local URI for global dereferenceability

In the system we need a mental (local) identifier and a way to retrieve its definite description, avoiding the problem of homonymy of names. URLs provide the needed local identification and dereferenceability. We create therefore a new local identifier called $SURL$ (for Semantified URL) with the following characteristics:

1. A $SURL$ represents a local description, perspective or mental model a particular person has with respect to a *real world entity*.
2. Each person has its own $SURL$, which is unique, for each *real world entity* known.
3. Given a $SURL$, we can retrieve a description (mental model) of an real world entity.
4. One $SURL$, representing a local descriptor, might refer to only one real world entity.
5. The SURL allows us to disambiguate homonymy in names used in the mental entities.

We call this identifier *SURL* and not just URL, because of the above-mentioned desired properties.

[5] which basically means that you have a referent, you just don't know who/what it is

9.2.1.2 SURI: Global URI for global identification

The properties of (global) uniqueness and persistence present in URIs makes them a good candidate for identifying real world entities (see Section 3.4.1). Permanence is another fundamental characteristic needed by a identifier for a real world entity, this is, in order to withstand the changes real world entities undergo over time[6].

A **Semantified URI** or $SURI$ is a machine processable identifier for real world entities (sort of the counterpart of names in mental entities) that allows us to uniformly, uniquely and universally identify real world entities while being permanent, i.e., resistant to its evolution. Other important properties are:

1. $SURI$s are shared among different local perspectives ($SURL$s).
2. in order to ensure global uniqueness of the $SURI$ as an identifier for the real world entity, the entity to which the $SURI$ is being assigned has to be also globally identifiable via its metadata EM (i.e., it cannot globally identify an entity that is not globally differentiable).
3. View Independence: a $SURI$ does not commit to any particular description of entity, i.e., given a $SURI$ it should be possible to retrieve all the possible diverse views.
4. Type Independence: a $SURI$ aims at solving the real world entity identifier problem without committing to any particular entity type.
5. Persistence: Once a $SURI$ is created for a real world entity, it must always exist to refer to that entity, i.e., it cannot cease to exit (only be merged and split).
6. Any framework for maintaining $SURI$s should minimize false negatives (FN) while strictly avoiding false positives (FP) [BLFM05]. False negatives (FN) are cases in which different $SURI$s are given for the same real world entity. False positives (FP) are cases in which the same $SURI$ is given for different real world entities.

9.2.2 Identifiers: definition

If we recall the entity definition presented in Model Object 10, an entity e is a tuple $e = \langle$ *ID, SURL, etype, name, EM,* $[link], [copy]\rangle$. The ID and $SURL$ are the digital counterparts for the real world identifier and the mental identifiers. In what follows we present the Model Objects for the identifiers.

> **Model Object 14** (Universally Unique Entity Identifier (ID)) ID is the entity identifier at the "real world" layer. The pragmatics of the ID lies in the identification process, i.e., if two entities refer to the same real world entity,

[6]Permanence is an important characteristics in Biometrics which measures how well a biometric resists aging and other variance over time `http://en.wikipedia.org/wiki/Biometrics`

then they should have the same ID, and the system must support operations (matching, alignment, propagation, merging, splitting) for achieving this.

There are two kinds of IDs depending on how well the metadata in EM identifies *globally* the *real world entity* being referred. We say that an entity is well identified if we can globally distinguish entities (of the same type) from one another (i.e., uniquely identify a real world entity based on the attributes in EM and the identifying sets, see Section 8.2.2). On the other hand, an entity is not well identified, if based on EM and the identifying set we cannot distinguish unequivocally to which real world entity it refers to. We will use the $SURI$ as ID for well identified entities, and the $SURL$ as ID for entities that are not well identified.

Model Object 15 (Semantified URI ($SURI$)) $SURI$ is the logical universal resource identifier (URI) of the entity which is used to identify uniquely an entity at the "real world" layer. Different entities created from different perspectives, even by different users, should have the same $SURI$ if they all refer to the same *real world entity* (see Section 9.2.1.2).

Model Object 16 (Semantified URL ($SURL$)) $SURL$ is the logical URL of the entity which is used to deference the metadata of a particular mental entity, i.e., it uniquely identifies a mental entity (as opposed to the name) and allows dereferencing the mental representation (the definite description). Each entity e has a different SURL, therefore, multiple entities referencing the same *real world entity* (identified by the same ID) will have different $SULR$s.

Given that the SURL represents a particular description of an entity from a particular perspective, this description (given by EM in e, see Model Object 10) might not be sufficient to globally identify a particular real world entity from other real world entities, we say in this case that EM does not identify well enough e, and as such, we cannot use the $SURI$ as ID. In such cases we will use the $SURL$ as ID of e.

We must highlight that **the purpose of the SURL is dereferenceability**, i.e., given a $SURL$, return the set of metadata that this particular mental model or perspective describes, about a *real world entity*. Whereas **the purpose of the ID (or SURI) is identification of entities to achieve interoperability**, i.e., if two entities have the same $SURI$, then they refer to the same real world entity, independently of where they are located. Furthermore, the $SURI$ encodes no particular mental model or perspective, i.e., given a $SURI$, there is not only one EM describing the real world entity attached to the $SURI$, but all the possible EMs (is up to the user to select one of them, or create another that fits his/her needs).

9.3 Dynamics in Entities

This section describes how we manage the life cycle of the identifiers of digital entities, complying with the desired properties of entities presented in Section 3.4.3 and their identities presented in Section 9.2.

In the following subsection we will adopt a simplified version the Model Object 10 to describe the entities assuming all entities being exemplified in the figures belong to the same $etype$, therefore the entity model becomes, $E = \langle SURL, ID, link, \mathsf{EM} \rangle$.

Based on the entity definition, we recognize three kinds of digital entities: original, copied and shared. Original entities are the entities created form scratch, copied entities are entities that we copied from already existing entities, copying (and possibly modifying) the metadata definition EM, and a shared entity is an entity that adopts (and is always synchronized with) the metadata definition EM from a source entity (which can be any of the three types) in different instance of the system, i.e., another knowledge base using the same architecture presented in Chapter 3. The following subsections will define how the basic operations (create, update and delete) over the entities will affect the identity of entities, i.e., their $SURL$s and $SURI$s, and how these identifiers are managed.

9.3.1 Original entity

We say that an entity is an original entity if it is created manually by a user, or by a batch process when bootstrapping the entities into the system, importing them from other sources (not based on our framework defined in Chapter 3) such as Yago[7] or Geonames[8].

9.3.1.1 Create entity

When an entity is created from manual user input, a new entity E is created locally in the concrete knowledge component of the knowledge base (see Section 3.4). Let us assume that a peer ($p1$) creates an entity, in this case it creates a new $SURL$ for the entity, the ID can take two possible values according to how well specified the entity is by the metadata given in EM (see Model Object 14 in page 100 in Section 3.4.3) and the identifying sets for its $etype$ (see Model Object 13 in page 89). The process of assigning the ID to the entity is triggered once the entity is fully created and all the attribute values that one is interested in have been defined, calling **Entity ID creation** method defined as follows:

Entity ID creation In order to assign the ID to an entity, we have to consider the following two cases, according to how well specified the entity is by the metadata given in EM and the identifying set:

[7] http://www.mpi-inf.mpg.de/yago-naga/yago/

[8] http://www.geonames.org/

1. **SURI as** ID: if EM provides enough information to make the entity identifiable, i.e., there are attribute values for at least one identifying set for its $etype$ and provided that a global search (in all the instances of the framework) did not find another matching entity that already contains a $SURI$, then a new $SURI$ can "safely" be create for the new original entity. Figure 9.1 a) shows E1 as an example new original entity with a $SURI$ as ID.

2. **SURL as** ID: if EM does not provide enough information to make the entity identifiable, i.e., no identifying set is filled in, then we use the entity's $SURL$ as ID. This is due to the fact that global search and entity matching will not be able to find other entities referring to the same *real world entity* that already contain a $SURI$ (or even $SURL$), given that entity matching relies on identifying sets[9]. Figure 9.1 b) shows E2 as an example new original entity with a $SURL$ as ID.

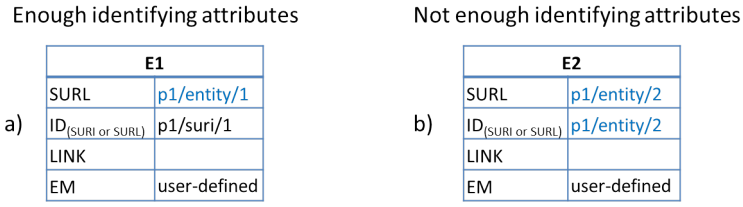


Figure 9.1: Entity creation: a) Entity with enough identifying attributes has a $SURI$ as ID; b) Entity with not enough identifying attributes has a $SURL$ as ID.

The ID can be set at any point in time after the entity is created and all the attribute values are assigned. As long as the entity remains for local use only, the ID is not essentially needed, given that the entity is locally identifiable and dereferenceable also by its $SURL$. The ID of an entity has to be set (in either of the above cases) before a peer is capable of making its entity public and therefore findable by other peers.

9.3.1.2 Update entity

Once the entity is created, we can update it by performing CUD (create, update, delete) operations on the metadata EM. The following subsections specify how each of these operations affects the identity ID of the entity.

Create attribute The following cases are considered:

[9] If the user is in the loop, we can search and suggest similar entities that are well identified (have a $SURI$) so that we promote the reuse of entities.

Entity ID is a SURI In the case the new attribute value does not fill in any new identifying set, the ID is not affected. If the new attribute creates a new identifying set a compatibility check (see Section 8.2) with other entities with the same $SURI$ is needed (as the entity might have been copied and updated somewhere else). If it is still **compatible**, the ID is not affected, if it is **non-compatible**, a new $SURI$ as to be created (see Section 9.3.1.1).

Entity ID is a SURL If the first identifying set is filled in by the creation of the new attribute value, then the Entity ID creation (see Section 9.3.1.1) method should be called to assign a new $SURI$ as ID. Figure 9.2 shows an example where the new attribute creates the first identifying set for $E2$ therefore assigning a $SURI$ to the updated $E2'$.

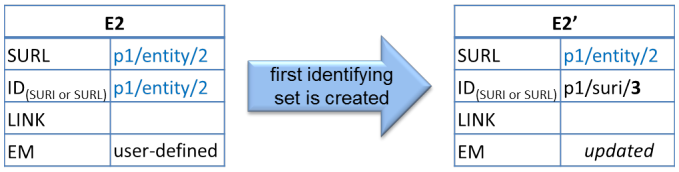


Figure 9.2: Updating an entity with $SURL$ as ID when creating the first identifying set gives a $SURI$ to the entity.

Update attribute The following cases are considered:

Entity ID is a SURI Given the principle of compatibility (see Section 8.2), if the updated version of the entity is **not compatible** with the original version, a new $SURI$ has to be assigned to the entity (see Section 9.3.1.1). If we are in the presence of the **undefined** compatibility, the user has to be notified and asked. Figure 9.3 shows an example where updating the identifying set of entity $E1$ triggers the creation of a new $SURI$.

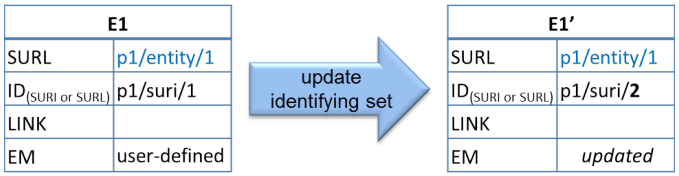


Figure 9.3: Updating an existing identifying set of an entity with $SURI$.

Entity ID is a SURL The ID does not change, as an identifying set cannot be created by changing the values of attributes.

Delete attribute The following cases are considered:

Entity ID is a SURI In the case of deleting an attribute value that is part of the last remaining identifying set, the $SURI$ has to be replaced by the $SURL$.

Entity ID is a SURL The ID does not change, as an identifying set cannot be created by deleting the values of attributes.

9.3.1.3 Delete entity

The entity is deleted locally, which means that the $SURL$ ceases to exist. If the $SURL$ is not used as value of a relational attribute locally, deleting an entity has no further local effect. If the $SURL$ is used as value of a relational attribute, then the reference to the $SURL$ is deleted when the systems tries to dereference it (read), i.e., it acts as a lazy delete cascade in a foreign key.

9.3.2 Copied entity

Let us assume a peer 2 ($p2$) finds in $p1$ an original entity s/he is interested in and decides to copy the entity locally in order to have information about this entity, e.g., the phone number or email address of a person or an organization, or to use it as value of relational attributes.

9.3.2.1 Create entity

Peer 2 ($p2$) creates a local copy of the found original entity assigning a new local $SURL$, copying (the desired subset of) the metadata[10] with the condition that at least one identifying set is copied whenever available, and assigning the ID according to the following cases:

1. if the original entity begin copied has a $SURI$ as ID, then the ID of the new entity is the same as the one in the original entity. Figure 9.4 shows a copy example of $E1$ from $p1$ to $p2$, creating a new local entity $E3$ with its own $SURL$ but with the same $SURI$ as $E1$.

   Note that the $SURI$ gets also propagated in any subsequent copy, as can be seen in Figure 9.4 when $E3$ gets copied to $E5$.

2. if the original entity being copied has a $SURL$ as ID, then the ID of the new entity is its own new local $SURL$. In the example of Figure 9.5 when $p2$ copies $E2$ from $p1$, it creates a local entity as defined by $E4$. The same applies when $p3$ copies $E4$ locally, creating E6.

[10] There is also an issue to be considered when the language and concepts used to described EM^1 are local to $p1$, but this is out of the scope of this chapter, for new we assume all the linguistic and conceptual part of the knowledge base is shared among $p1$ and $p2$. In Chapter 7 (Knowledge propagation) we deal with the case when there is a difference in the knowledge of different interacting peers.

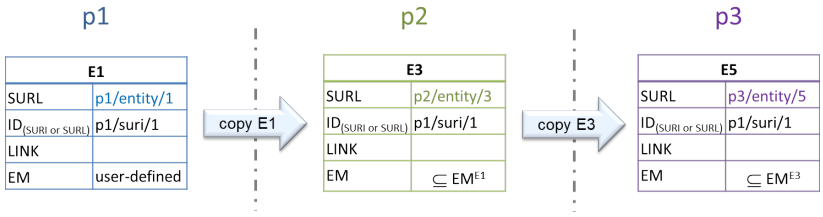


Figure 9.4: Copying an entity that has a $SURI$ from p1 to p2, and from p2 to p3

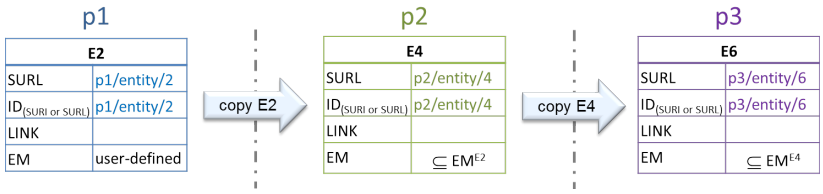


Figure 9.5: Copying an entity that has as ID a $SURL$ from p1 to p2, and from p2 to p3

Note that in this case there is no relation between entities $E2$, $E4$, and $E6$, other than some common metadata in EM, as their IDs are all different. Furthermore, from the point of view of $p3$, it is as if $E4$ was an original entity created by $p2$, as opposed to the case with a $SURI$ as ID in Figure 9.4 where $p3$ knows that $p2$ is not the generator of the $SURI$ of $E3$.

9.3.2.2 Update entity

In this section we will define the effects of updating the copied entities.

Create attribute The following cases are considered:

Entity ID is a SURI ($E3$ from Figure 9.4) In the case the new attribute value does not fill in any new identifying set, the ID is not affected. If the new attribute creates a new identifying set, a compatibility check (see Section 8.2) with other entities with the same $SURI$ is needed (at least the original entity if still exists). If it is still **compatible**, the ID is not affected, if it is **non-compatible**, a new $SURI$ as to be created (see Section 9.3.1.1).

Entity ID is a SURL if the first identifying set is filled in by the creation of the new attribute value, then the Entity ID creation (see Section 9.3.1.1) method should be called to assign a new $SURI$ as ID.

Update attribute The following cases are considered:

Entity ID is a SURI ($E3$ from Figure 9.4) Given the principle of compatibility (see Section 8.2), if the updated version of the entity is **not compatible** with the original version, a new $SURI$ has to be assigned to the entity (see Section 9.3.1.1). If we are in the presence of the **undefined** compatibility, the user has to be notified and asked.

Entity ID is a SURL ($E4$ from Figure 9.5) nothing happens, as an identifying set cannot be created by changing the values of attributes.

Delete attribute The following cases are considered:

Entity ID is a SURI In the case of deleting an attribute value that is part of the last remaining identifying set, the $SURI$ has to be replaced by the $SURL$.

Entity ID is a SURL nothing happens, as an identifying set cannot be created by deleting the values of attributes.

9.3.2.3 Delete entity

The entity is deleted locally, which means that the $SURL$ ceases to exist. If the $SURL$ is not used as value of a relational attribute locally, deleting an entity has no further local effect. If the $SURL$ is used as value of a relational attribute, then the reference to the $SURL$ is deleted when the systems tries to dereference it (read), i.e., it acts as a lazy delete cascade in a foreign key.

9.3.3 Shared entity

A peer might also want to **share** the definition of a found entity, fully relying on the metadata of the source entity. This means that any update on the original entity will be seen (propagated) to the shared entity, i.e., both entities will be synchronized. These updates include not only the metadata definition, but also the updates in the ID. The source entity can be any of the three types, and original entity, a copied entity, or even another shared entity.

If the original entity ceases to exits, then a copy-like procedure (following the rules of copy in Section 9.3.2.1) is performed, where the metadata EM is taken from a local cache (see Section 9.3.3.1) of the metadata of the original entity.

9.3.3.1 Create entity

Given that the source entity can be any of the three types, and original entity, a copied entity, or even another shared entity, here we will exemplify three possible cases:

1. Original entity → Share entity. In this case we will see what happens when the source entity is an original entity. Note however that when the source entity is a copied entity, the procedure is the same as if it was an original entity, given that the metadata EM is contained in the copied entity. Therefore the "Copied entity → Share entity" case is the same as this case.

2. Original entity → Share entity → Share entity. In this case we will see how the re-sharing of an entity, i.e, the source entity is a shared entity, affects the ID.

3. Original entity → Share entity → Copy entity. In this case we will explore what happens when a shared entity is later copied.

Original entity → Share entity When sharing an entity, a new entity is created locally with a new $SURL$, no metadata is stored in the local entity (but is cached in the peer), the ID is the same as the original entity being shared, and the $link$ attribute takes the $SURL$ of the original entity (in order to know where to retrieve the metadata from). Figure 9.6 shows the sharing process of $E1$ (the original entity) from $p1$ from $p2$, creating $E7$.

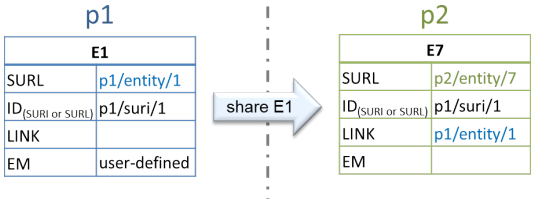


Figure 9.6: Sharing an entity that has a $SURI$ from p1 to p2

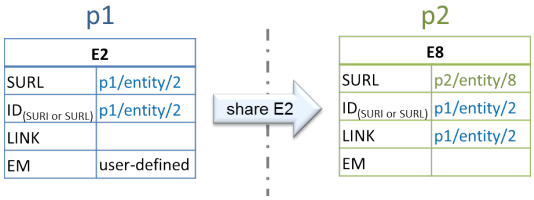


Figure 9.7: Sharing an entity that has as ID a $SURL$ from p1 to p2

Figure 9.7 shows the same sharing process for an original entity that has a $SURL$ as ID. In this case the $SURL$ as ID gets propagated (as opposed to the **copy** case in Section 9.3.2).

Original entity → Share entity → Share entity Figure 9.8 shows the sharing process of $E1$ (the original entity) from $p1$ to $p2$, and from $p2$ to $p3$. The new local

shared entity in $p3$ has the same ID as the previous two, but the $link$ attribute does not point to $E7$'s $SURL$, but to $E1$'s $SURL$, given that the metadata $p3$ decides to trust is actually $E1$'s, which is the original entity (the one with the metadata).

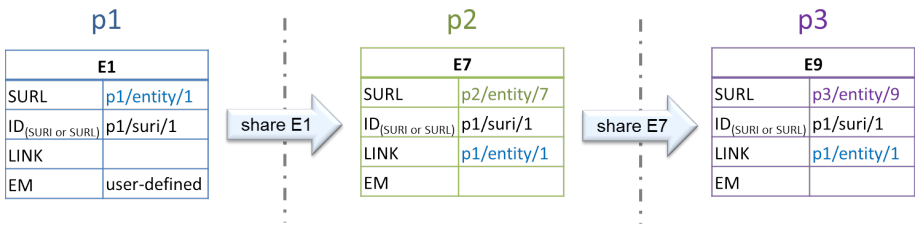


Figure 9.8: Sharing an entity that has a $SURI$ from p1 to p2, and from p2 to p3

Figure 9.9 shows the transitive sharing process of $E2$ (the original entity that has a $SURL$ as ID) from $p1$ to $p2$, and from $p2$ to $p3$. In this case the $SURL$ as ID gets propagated from $E2$ to $E8$ (as opposed to the **copy** case in Section 9.3.2). Note also that the $link$ attribute in $E10$ also points to $E2$ (and not to $E8$) as $E2$ is the original entity that contains the metadata.

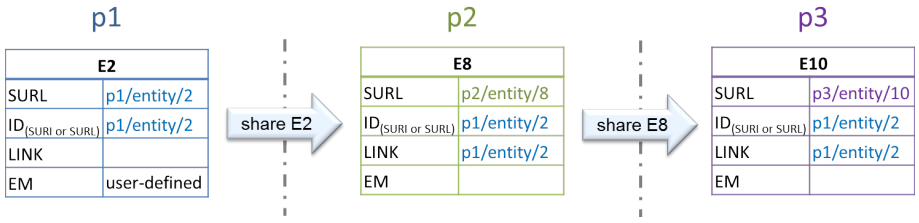


Figure 9.9: Sharing an entity that has a $SURL$ as ID from p1 to p2, and from p2 to p3

Original entity → Share entity → Copy entity Copying a shared entity is as if the original entity was being copied.

Figure 9.10 shows the sharing process of $E1$ (the original entity) from $p1$ to $p2$, and later copying the shared entity $E7$ from $p2$ to $p3$ to create $E13$. The creation of $E7$ is the same as described in Section 9.3.3.1. When copying a shared entity the new local copied entity in $p3$ has the same ID as the previous two, but the $link$ attribute is empty (as it should by the copy rule specified in Section 9.3.2.1). The final outcome of copying a shared entity is as if $E13$ was copying the original entity, as the shared entity $E7$ is just a cached version of the original entity containing no metadata EM on its own (besides what is cached).

Figure 9.11 shows the sharing process of $E2$ (the original entity that has a $SURL$ as ID) from $p1$ to $p2$, and later copying the shared entity $E8$ from $p2$ to $p3$ to create $E14$. The creation of $E8$ is the same as described in Section 9.3.3.1. The ID of $E14$ is its own $SURL$, as the entity being copied does not have a $SURL$,

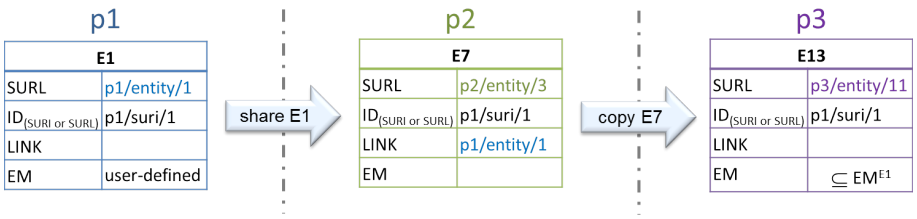


Figure 9.10: Sharing an entity that has a $SURI$ from p1 to p2, and later copying the shared entity from p2 to p3.

and the metadata EM is taken from the original entity (or the cached version of the metadata EM in $E8$).

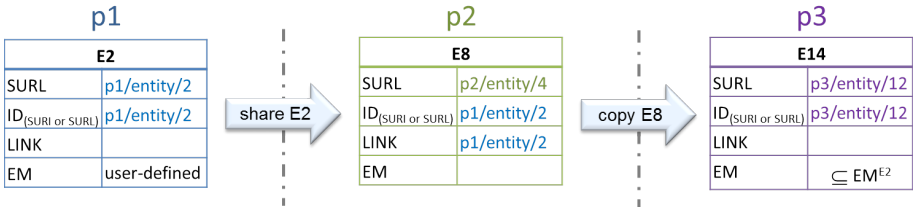


Figure 9.11: Sharing an entity that has as ID a $SURL$ from p1 to p2, and later copying the shared entity from p2 to p3

Caching metadata In order to avoid dereferencing the entity being shared by the $SURL$ value in the $link$ attribute to retrieve its metadata each time the system needs to shows it to the user or perform an operation with it, the system creates a cache version of the metadata of the original entity. This caching needs to consider that the original entity can be updated, and therefore the system needs to check whether the metadata in the cache is up to date. This can be achieved by creating a hash signature (e.g. MD5) over the cached metadata, and request the hash signature of the metadata of the original entity; if these two hashes match, then the cache is up to date, if not, it needs to be updated. Another option is to encode versioning information in the entity definition.

Also, an RSS like mechanism can be implemented, where the peer with the original entity makes public the data, and the ones linking to it subscribe to this feed.

9.3.3.2 Update entity

When updating the metadata of a shared entity, i.e, **creating**, **updating** and **deleting** attributes and attribute values, the metadata EM of the original entity gets copied locally applying the rules for copying entities as described in Section 9.3.2.1, followed by the rules of updating copied entities described in Section 9.3.2.2.

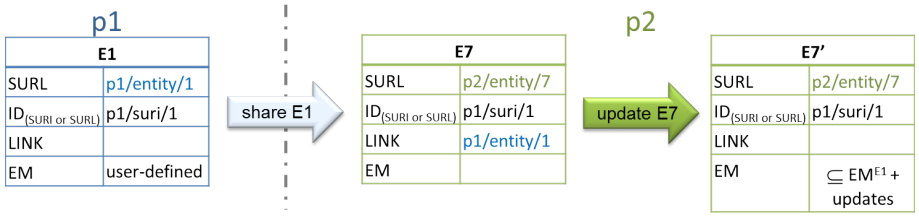


Figure 9.12: Updating a shared entity for which the original entity has a $SURI$. The update operation in the example did not change the ID, for example, more non-identifying attributes were added.

Figure 9.12 shows an example where $p2$ updates $E7$ which was previously shared from $E1$. The updated $E7$' preserves the same ID (since by the copy rule as $E1$ has a $SURI$ as ID, the copied entity contains the same ID), the metadata EM is copied to $E7$' and $p2$ updates it as needed. After this copy-like procedure, the rules for updating entities presented in Section 9.3.2.2 are applied. The update operation in the example of Figure 9.12 did not change the ID, for example, because more non-identifying attributes were added to the updated $E7$'.

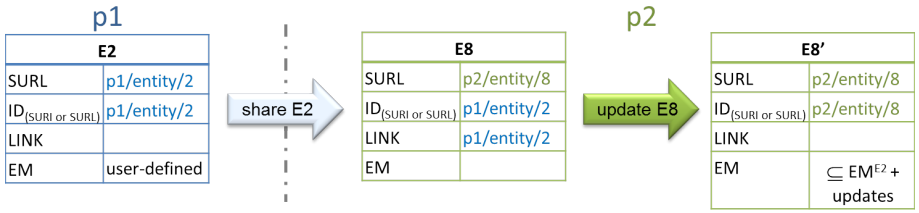


Figure 9.13: Updating a shared entity for which the original entity as a $SURL$ as ID. The update operation in the example did not change the ID, for example, a value was changed.

Figure 9.13 shows an example where $p2$ updates $E8$ which was previously shared from $E2$. The updated $E8$' assigns as ID its own $SURL$ (since by the copy rule as $E2$ has a $SURL$ as ID, the copied entity uses as ID its own $SURL$), the metadata EM is copied to $E8$' and $p2$ updates it as needed. After this copy-like procedure, the rules for updating entities presented in Section 9.3.2.2 are applied. The update operation in the example of Figure 9.13 did not change the ID, for example, because a value was changed.

9.3.3.3 Delete entity

Similarly to the deletion of an original entity, when a shared entity is deleted locally, which means that the $SURL$ ceases to exist. If the $SURL$ is not used as value of a relational attribute locally, deleting an entity has no further local effect.

If the $SURL$ is used as value of a relational attribute, then the reference to the $SURL$ is deleted when the systems tries to dereference it (read), i.e., it acts as a lazy delete cascade in a foreign key.

9.4 Entity alignment

When we detect that two entities have different IDs but refer to the same *real world entity* (via entity matching outlined in Chapter 8 or by manual user input) we need to align the entity metadata and unify the two different IDs into one. This difference in ID can be due to the partiality of knowledge or the dynamic nature of entities that evolve (change attributes and values) over time. This service is required to comply with the $SURI$ property 6 "minimize False Negatives" of Section 9.2.1.2.

9.4.1 Aligning metadata

Merging the metadata definition EM of two different entities should take into account the following cases:

1. New attribute and value: in this case we can automatically add the missing attribute, provided that when adding the attribute, the resulting EM is compatible with both source entities. If this process of alignment is triggered after the entity matching is successful, there should be no issue of compatibility. The compatibility issue could be triggered when the process was initiated due to a manual entity alignment request by a user, in which case the user should also be requested to solve the possible compatibility issue.

2. Existing attribute definition, with different attribute value: we can have two cases:

    2.1. if the attribute definition is a set, then we can add the value with no problem, e.g., a phone number.

    2.2. if the attribute definition is not a set, then we need to resolve the difference. This case normally requires the intervention of the user. However, in the case the domain D of the attribute definition AD (see Model Object 9) is a concept or an entity, we can use semantic matching (see Chapter 5) or compatibility on metadata (see Chapter 8) respectively to compare the values and try to resolve the inconsistency automatically.

When comparing concepts, we need to study the compatibility of attribute values, for example: more general vs. less general values. Also, values in a close range for integers, reals, dates and coordinates could also be compared and resolved semi-automatically (asking for the confirmation of the user). Resolving automatically these ranges, however, is highly application domain and entity type

dependent. For example, is not the same having a difference of 20 centimeters in the definition of the height of a mountain, which can be considered as a precision difference and therefore ignored, to having the same 20 centimeters difference in the height of a person, which cannot normally be ignored.

9.4.2 Unifying identifiers

As a rule of thumb, we define that the ID of the more specified entity (i.e., the one that has the most identifying attributes defined) will be preferred, and if there are differences in the attributes, the system should suggest to merge them. The following sections will analyze the different combinations according to the kind of ID of the entities being merged.

9.4.2.1 SURL to SURL

In order to unify two entities that have $SURL$s as IDs one entity has to share (link) to the other, as the only way the ID of an entity can have a $SURL$ as value is by sharing (see Section 9.3.3.1). The user that decides to take the SURL of the other user loses the metadata of the entity which will be shared.

Figure 9.14 shows an example of two entities ($E14$ and $E2$) from two different peers ($p4$ and $p1$) that are found to refer to the same *real world entity*, which are later unified. $E2$ shares with E14, therefore the $link$ and ID attributes in $E2$ take $E14$'s $SURL$ as value, and the metadata is now taken from $E14$.

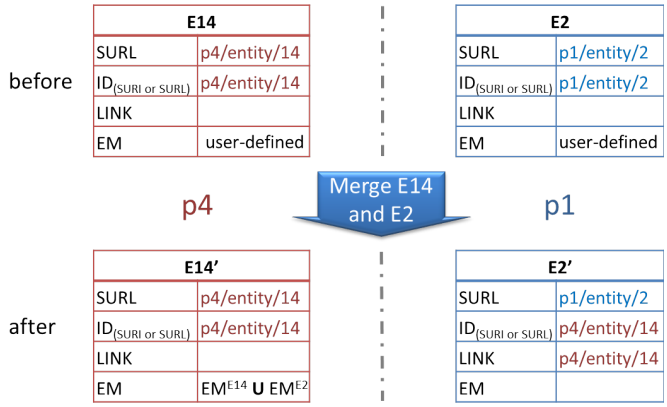


Figure 9.14: Unifying two entities that have $SURL$s as IDs. In this example $E14$ becomes the original entity and $E2$ shares from $E14$.

Assuming $E2$ was shared with other peers as in the case of Figure 9.9, there will be a set of entities $\{\underline{E2}, E8, E10\}$ where $E2$ is the original entity whose $SURL$ is used as ID; we will call this a $SURL$ family. If we unify E2 and E14, as in Figure 9.14, we need to propagate the change in the ID and $link$ attributes from $E2$ to $E8$ and $E10$ setting the ID and $LINK$ attributes to point to

$E14$ (similarly to $E2$). Figure 9.15 shows the entities before the update and after the update, showing how the whole $SURL$ family is updated and unified.

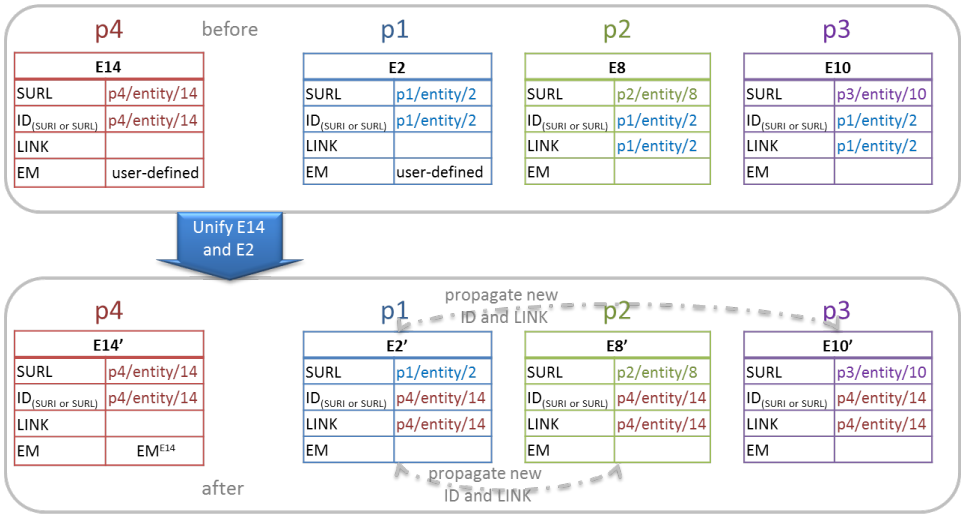


Figure 9.15: Unifying entities with 2 $SURL$s, one of which as been shared. Before: the entities with different $SURL$s; After: all the entities with the same $SURL$s as ID

9.4.2.2 SURI to SURL

In order to unify two entities, one of which having a $SURI$ and the other a $SURL$ as ID, the entity that has the $SURL$ has to import at least one identifying set from the entity that has the $SURI$ (i.e., **an entity cannot have a** $SURI$ **as** ID **without an identifying set**). In this case, the $SURI$ will replace the $SURL$ as ID.

Figure 9.16 shows the case when we unify an entity that has a SURI ($E15$) with an entity that has a SURL ($E2$), which is being linked (shared) by other entities (as shown in the before part of the image $\{\underline{E2}, E8, E10\}$ corresponding to Figure 9.9). After the unification $E2''$ takes the $SURI$ from $E15$, and later the change in ID is propagated from $E2''$ to $E8''$ and $E10''$. The figure shows how the process of changing the ID in an original entity that has a $SURL$ as ID gets propagated from peer to peer.

9.4.2.3 SURI to SURI

The fact that two entities have different $SURI$s, but refer to the same *real world entity* might be due to two reasons:

1. A previous mistake, where a new $SURI$ was assigned even though another $SURI$ existed for the *real world entity* being referred. Maybe because the

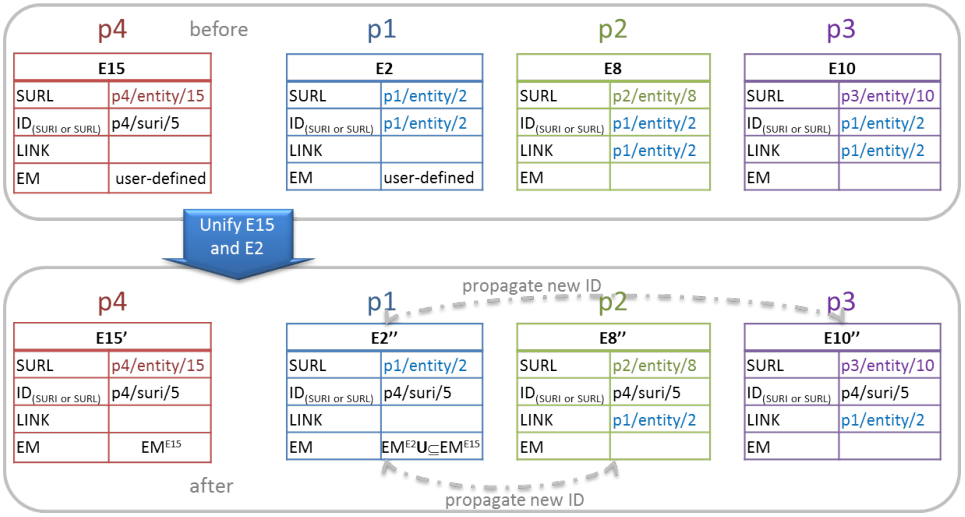


Figure 9.16: Unifying an entity with a $SURI$ with another with $SURL$ as ID that is shared.

global search in all the instances of the framework failed to retrieve the existing $SURI$ due to a variation in the name of the entity (e.g., Napoleon vs. Napoleone. Trento vs. Trient).

2. Different (disjoint) identifying attribute sets in both entities, which meant that entity matching failed.

In order to unify two entities with different $SURI$s, both entities have to share at least one identifying set, and one of the peers have to give up the $SURI$ of its entity. This means that the entity in the peer that gives up the $SURI$, the one that is modifying its entity has to import the minimal set of metadata from the other entity from where the new $SURI$ is being taken, in order to allow subsequent entity matching (that rely on identifying sets) to succeed (see Chapter 8).

Figure 9.17 shows an example of two entities with different $SURI$s, $E1$ and $E15$ that are found to refer to the same *real world entity* and therefore will be unified. In the example figure, the $SURI$ of $E1$ is already propagated to other peers via share to the entities $E7$ and $E9$ (as in Figure 9.8). If $p1$ decides to unify $E1$ with $E15$, then $E1$'s $SURI$ is updated to that of $E15$, and later this ID change propagated to $E7$ and $E9$.

If E1 was also copied to other entities, as in the case of Figure 9.4, then given that each copy contains its own metadata, they would need to start the unification process independently to try to merge the needed identifying sets, and acquire the new $SURI$. Note that a change in the $SURI$ of $E1$ does not affect the $SURI$ of $E3$ or E5 in Figure 9.4. Both $E3$ and $E5$ continue to refer to the same (originally referred) real world entity.

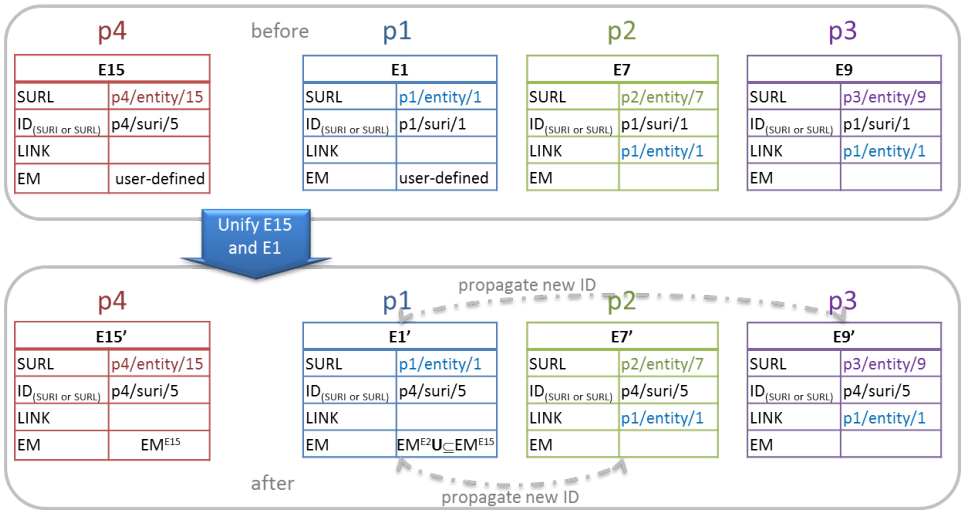


Figure 9.17: Merging an entity with a $SURI$ with another with $SURL$ as ID that is shared.

9.5 Related work

The *Internet Engineering Task Force* (IETF), together with the *World Wide Web Consortium* (W3C) are the main driving forces toward the standardization of identifiers of WWW resources. The IETF has released several standards that specify the format of digital identifiers such as URIs [BL94, BLFM05], URNs [DvGIF99], URLs [BLMM94] and IRIs [DS05]. URIs and URNs are simple and extensible means for identifying an abstract or physical resource [BLFM05], while the URL in addition to this, provides a means of locating and dereferencing the resource. IRIs are URIs with possible international characters (accents, ...). These standards, however, do not specify the conditions under which a new identifier should be created, and confusion is usually present on whether a particular identifier can also be created for non-www resources, or a www resource talking about it.

The W3C has also published several formats and best practices related to identifiers such as CURIE or compactURI[11] that defines a generic, abbreviated syntax for expressing URIs and Cool URIs for the Semantic Web[12].

Cool URIs for the Semantic Web [SC08] introduces a useful guidance on how to use URIs for things that are not Web documents using the HTTP protocol (non-HTTP URIs are not covered). The basic idea is to avoid confusion between an identifier for a real world object and the identifier of a web document describing the real world object, i.e, one URI cannot represent both. The solution is to use a special HTTP status code, "303 See Other", to give an indication that the requested resource is not a regular Web document, with a list of the location of

[11] `http://www.w3.org/TR/2010/NOTE-curie-20101216/`
[12] `http://www.w3.org/TR/2008/NOTE-cooluris-20080331/`

documents containing a description of the referred real world object. By doing so, it avoids ambiguity between the original *real world object* and resources representing it. This redirection can be technically implemented by using a Persistent Uniform Resource Locator (PURL)[13] that provides a level of indirection which, when retrieved, results in a redirection (e.g., via 301, 302, 303 HTTP status codes) to other locations for the final resource.

Given the proliferation of domain dependent applications, together with a lack of understanding on the difference between the URI of the real world object and the identifiers of the documents describing it, there has been a widespread creation of independent URIs for identifying the same real world objects. Several approaches [HHD07, JGM08, BSG07] try to overcome this issue. Initially the problem were heterogeneous descriptions of the real world objects, which is an intrinsic property of our world and as such cannot be avoided, see Section 3.4.2 (Mental entities), and for which much research is devoted. However, now we are faced with heterogeneous identifiers, which is not an intrinsic property of the world, but one introduced by a lack of proper understanding of the difference between real world and mental models, and the plethora of existing identification schemes (URNs, OkkamIDs [BSG07], DOI[14], ORCID [FGT11], ResearcherID[15], Open ID[16], WedID[17], DUNS system[18], Organization ID[19], to name a few).

Hogan et al. [HHD07] propose the creation of a canonical URI for entities by fusing identifiers based on "inverse functional properties" specified in Web Ontology Language (OWL) descriptions [MvH04]. For example, foaf:mbox[20] is defined to be unique to a person, therefore, if two instances share the same foaf:mbox property, then it could be inferred that both refer to the same real world entity. However, one needs to be careful when selecting the canonical URI, and whether this URI actually refers to the real world object being identified, or it is yet another representation of the object at hand.

A widespread approach used by the Semantic Web community relies on the owl:sameAs relation between entities that are considered to be the same; this is the approach adopted by DBpedia [ABK+08]. However, the semantics of owl:sameAs predicate defines that the URIs linked with it have the same identity [DS04], i.e., must be exactly the same resource with respect to all properties (the same perspective). The major drawback of this approach is that the two URIs become indistinguishable even though they may refer to different perspectives of the same entity according to the context in which they are used [JGM08]. This implies that there is no clear separation between the identifier of a set of properties (or description) of

[13] `http://purl.oclc.org/docs/index.html`
[14] `http://www.doi.org/`
[15] `http://www.researcherid.com/`
[16] `http://openid.net/`
[17] `http://www.w3.org/2005/Incubator/webid/spec/`
[18] `http://en.wikipedia.org/wiki/Data_Universal_Numbering_System`
[19] `https://www.arin.net/resources/request/org.html`
[20] The email address property defined in FOAF`http://xmlns.com/foaf/spec/`.

an entity, and the URI of the entity itself, as recommended in the W3C Note "Cool URIs for the Semantic Web" [SC08].

Jaffri et al. [JGM08] describe an architecture for managing URI synonymity by using Consistent Reference Services (CRS). The CRS is an independent knowledge base that contains the knowledge about URI synonyms. Equivalent URIs are grouped into "bundles" which are themselves given their another URI. A service in the CRS can be queried for finding equivalent URIs and for retrieving the corresponding bundle. The approach does not include entity deduplication algorithms, but relies on already existing links between entities on the web (e.g., owl:sameAs predicates); however, as stated before, the semantics of this predicate is not always well understood and therefore often misused. Another consideration with this approach is that while trying to solve the multiplicity of URIs already created to refer to entities, they do introduce yet another URI for the bundles.

The Linked Data Integration Framework (LDIF) [ASIBB11] is designed to translated different data sources from the Web of Lined Data into a single clean local target representation while keeping provenance information. The Identity resolution module relies on user-defined matching heuristics in the Silk Link Discovery Framework [IJB10] to find synonymy in URIs from different data sources referring to the same real world object. Once the duplicates are identified, all URIs are replaced by a single URI and an owl:sameAs statement is added between the selected URI and the original one. The main drawback of the approach is that whenever new sources of information are available, or he information is updated, the whole importing process has to be performed in order to import the new data, including new possible heuristics to compare entities. While the approach is valid, there are cases when a centralized approach is not possible, and reasoning has to be performed using dynamic data in open environments.

The Okkam framework [BSG07, BSB08] aims at providing entity identifiers for the Web following a proprietary URI scheme. Okkam identifiers guarantee uniqueness across space and time, which prevents generation of duplicates. Entities are untyped, i.e., they are not described with a fix schema. This makes the management of the entity identifier application independent and allows for the creation of Okkam IDs for any kind of entities, a process called *okkamization*. The framework provides services for adding new entities and creating their identifiers, adding/modifying/removing descriptions to existing entities and searching for entities already existing in their knowledge base.

While the Okkam definition for entity is close to our understanding of entities; and the aim of providing a global, unique and persistent identification of real world entities is much similar to our definition and aim, there are several key points that differentiate our approaches. First, there is no mention in the architecture on how to manage the multiple and diverse local views each application domain has about the same real world entity, i.e., there is no support for the differentiating between the global identifier and the mental models (which is a desired property as stated in [SC08]), there is only one central repository with all the information and the descriptions are all stored inside one entity. Creating a new URI for an entity normally

implies that the issuer has knowledge about the entity that wishes to make public. It is not clear how it can be accurately determined that the Okkam URI commits to the same entity to which a knowledge provider wishes to refer [JGM08]. Second, we argue that the entity type can provide much information on how to effectively compare descriptions about particular entities, e.g., it is not the same having a difference of 20 centimeters in the height of a mountain and that of a person, one can normally be neglected, the other cannot. Third, the possibility of accommodating to several (future) application domains is solved by having untyped entities. On the contrary, we provide a framework for any application domain to develop their own entity types, and therefore exploit their domain expertise. However, if the domain experts do not want to create their entity types, we provide a basic (root) entity type, which by default is assigned to an entity, called $Entity$, similar to the *Thing* root schema in schema.org.

There are several approaches for identifying entities that are domain dependent. For example, the Open Researcher & Contributor ID (ORCID) [FGT11] initiative aims at solving the author name ambiguity problem in scholarly communication by establishing a global, open registry of unique identifiers for researchers. ResearcherID also assigns unique identifiers to authors in the research community. The WedID and Open ID frameworks allow the identification of people as users on the Web. Both protocols enable secure authentication on the Web. An Organization ID (Org ID) represents a business, nonprofit corporation, or government entity in the American Registry for Internet (ARIN) database. The DUNS system identifies organizations, specially banks, or any other form of business that has a distinct legal identity.

The digital Object identifier (DOI) is a system for identifying objects, normally intellectual content, in the digital environment. DOI names are not enforced to be URIs nor it is registered as a URN, despite complying with all the requirements for being a URN. According to the DOI definition, the system differs from other standard registries such as the International Standard Book Number (ISBN) and the International Standard Recording Code (ISRC) in that DOI names can be associated with services, such as obtaining well-structured metadata by using the identifier, as opposed to just being a format or numbering scheme for creating identifiers. While there is a widespread use of DOI identifiers in the scientific community, our main concern is its applicability to only a particular domain, and the fact of being a centralized system committing to a single set of metadata, a single point of view.

9.6 Summary

In this chapter we presented an Identity management framework that captures the semantics of real world entities and mental entities (local perspectives or views about the real world entities). The main novelty is precisely in being able to distinguish between the identifiers of these two kinds of entities. We have seen in the related work that managing the identity of entities is a much needed capability in

the current open web of data. We have listed some approaches that try to solve the issues, and in fact, to some extent go in the right direction. For example, the work in CRS [JGM08] highlights the need of maintaining different identifiers for each local views or descriptions, and OKKAM [BSB08] aims at providing a unique type and domain independent global identifier referencing the real world entities.

In our model, we have recognized the importance of both kinds of identifiers, as the only way for a digital system to interact with real world entities, identified by what we define as $SURI$, is by means of local (possibly diverse) views provided by people, identified by what we define as $SURL$. We have defined how the maintenance of the entities (create, update and delete operations) affect the both local and global identifiers. We also define mechanisms to merge identifiers, whenever it is recognized that different identifiers actually refer to the same real world entity.

The importance of this chapter does not lie in the format of the identifiers themselves, but in the definition of how the lifecycle of digital entities (representing real world entities) affect the digital identifiers, $SURI$ and $SURL$, in a distributed and heterogeneous environment, preserving the globality, uniqueness, permanence, type and view independence of the global digital identifier of the real world entity.

We have not committed to any particular format or numbering scheme for the $SURI$. Any format that complies with the basic URI properties and that also conforms to the $SURI$ properties presented in Section 9.2.1.2 can be adopted. For example, while complying with several URI properties, the Italian Fiscal Code[21], an Italian wide personal and legal identifiers, is not resistant to changes in the name of a person as it includes the first letters of all the names and last names of the person, i.e., if a person changes name, then the Fiscal Code also changes. To the best of our knowledge, the Okkam identifiers are the most similar identifiers, despite the outlined issues (see Section 9.5), that could be used as $SURI$; namely because its domain independence.

[21] `http://it.wikipedia.org/wiki/Codice_fiscale`

Part V

Evaluation

Chapter 10

Evaluation of Structure Preserving Semantic Matching

10.1 Introduction

In Chapter 5 we presented a semantic matching variation called Structure Preserving Semantic Matching (SPSM). SPSM takes as input two tree-like structures and produces a set of correspondences between the nodes and a numerical similarity score between $[0..1]$ while preserving certain structural properties. SPSM is based on the theory of abstraction and the tree edit distance algorithm.

The algorithm aims at automatically solving the semantic heterogeneity problem and is applicable to the web service integration problem, schema matching problem, and can also be used to compare etype definition in the Schematic Knowledge presented in Chapter 3, Section 3.3.

In this Chapter we present an synthetic evaluation methodology for SPSM inspired in the Ontology Alignment Evaluation Initiative (OAEI) [EIM+07] that consist in generating alterations of the trees given a set of original trees. We apply syntactic and semantic alterations to the original tress in order to test how well SPSM performs in the presence of these types of alterations. We then compare SPSM to state-of-the-art syntactic matching a semantic matcher based on Wordnet. Our results show that SPSM always performs better than any of the individual matchers, and that it is robust to the syntactic and semantic variations when applied separately and also in combination.

The rest of the Chapter is structure as follows: Section 10.2 presents the synthetic methodology for the evaluation of SPSM and in Section 10.3 we present the results when applying syntactic, semantic and combined alterations to original trees. Finally, Section 10.4 summarizes the Chapter.

Acknowledgment: The work presented in this chapter has been carried out with Paolo Besana, Lorenzo Vaccari and Pavel Shvaiko. Parts of this Chapter have been published in [GMY+08] and [VSP+12].

10.2 Methodology

We adopt a synthetic approach for evaluating SPSM which consist in deriving a tree-like structure from another original one using different kinds of operations in order to the change the syntax and the semantics of the original tree[1]. This is a common practice in Ontology and Web Service evaluation schemes such as the Ontology Alignment Evaluation Initiative (OAEI) and others [EIM+07, LSDR07, CEH+08, EFH+09, EFM+10]. Differently from the OAEI campaign, we generated the original trees automatically and then apply syntactic and semantic alterations to the original tree to create an altered tree. While altering the original tree, we compute the expected similarity score based on the types of alterations the original tree undergoes.

10.2.1 Tree creation

We have generated automatically 100 trees of depth 1 (i.e., a root and all child nodes being leaves). The rationale for selecting this depth was that given that SPSM is tailored for matching schemas, entity types definitions or web services, all of these normally consist on one parent node, and several immediate children. In the Web Service case the parent node is the function name and the parameters of the service are the children; in the schema case, normally the table name is the root node and the columns are the child nodes. The average number of nodes per tree was 8.

The labels of the nodes are composed of a random number of words selected from 9000 words extracted from the Brown Corpus[2]. We chose this corpus in order to avoid the bias towards Wordnet, from where our Background Knowledge was initially populated. The average number of words per node label was 4. This number was decided based on our experience with the analysis of labels in Directories such as Dmoz, Google and Yahoo in the Taxme2 dataset [GYAS09].

10.2.2 Tree alteration

In order to produce the altered trees that were to be matched against the original ones, we devised two types of alterations: *i)* syntactic alterations aimed at changing the content of the labels and the structure of the tree, without committing to maintaining the meaning (semantics) of the original tree; *ii)* semantics alterations aimed altering the original tree preserving, to some extent, the original meaning of the tree.

Both alterations aim at testing different part of SPSM. Semantic alterations tries to asses how much change in meaning can SPSM deal with, and syntactic alterations aim at testing if the matcher is resistant to misspellings or other changes

[1] We use trees as the input of the SPSM algorithm are always automatically converted into lightweight ontologies which have a tree-like structure.

[2] `http://icame.uib.no/brown/bcm.html`

in the labels. Each alteration can be controlled parametrically, i.e., the probability of effectively applying the alteration is a parameter in the system.

10.2.2.1 Syntactic Alterations

We define three levels of syntactic changes to the tree; *i)* introduce syntactic changes in the word (i.e., reproduce misspellings); *ii)* completely alter a word, by replacing it or dropping it; and *iii)* completely alter a node, by replacing the whole label with an unrelated one, or deleting the node.

Alter a word syntactically This alterations introduces misspellings to a word in the label. The words inside a label are tokenized using the space as separator. We know this is the correct separator given that we have created all the trees by the process described in Section 10.2.1.

First, given the configured probability, we decide whether to apply the alteration or not. Then, we select a set of words that will be altered, which preserves the given probability, and finally we select how to modify the words using three types of alterations: drop, add, change or swap a character in the word. The maximum number of alterations in a word is never greater that the total number of characters in the word and depends on a configuration parameter which indicates how much to change a word. For example, 100 would mean change the word completely, and 20, would change 1 letter in a five character word.

Alter a label syntactically This alteration adds or removes a word in the label of a node. We drop the word only if the label contains more than one word. The word added is randomly selected from the Brown Corpus, i.e., the same set used to create the original trees.

Alter a node syntactically There are three possibilities for this alteration:

1. replace a label in the node with an unrelated label. The unrelated words were taken from the Moby Thesaurus[3] to avoid bias towards Wordnet. The purpose of this alteration is to check whether the matcher can detect such drastic change, which would mean a lower similarity.

2. completely remove the node from the tree, which should decrease the similarity score between the trees.

3. change the order of the node between the siblings, which should not alter the similarity score between the trees.

[3] `http://icon.shef.ac.uk/Moby/`

10.2.2.2 Semantic Alteration

The goal of this alteration is to test the robustness of the matcher in presence of semantic alterations. We defined the semantic alteration as the replacement of a word in the label of a node by another word related to it, which can be more-general, less-general or equivalent (a synonym). The related words were extracted from the Moby Thesaurus and Wordnet using the synonym, hyponyms and hypernyms relations. Only one word in a node label is replaced.

10.2.3 Expected Similarity

When altering the trees we can compute the expected similarity given the known alterations. This allows us to compute the ground truth in order to asses the matching quality using standard measures such as precision, recall and f-measure [ES07]. The expected similarity score was computed using equation 10.1.

$$ExpSim(oT_i, aT_i^j) = \frac{\sum_{k \in N} score_k}{N} \tag{10.1}$$

Where i is the i*th* original tree oT, j is the j*th* altered tree aT of the i*th* original tree, k is the node number in both oT_i and aT_i^j. The expected score is normalized by the number of nodes of the original tree (N).

Initially, before applying any alteration, each $score_k$ is set to one since all the nodes of oT_i and aT_i^j are the same. Note that in this case, the *ExpSim* is 1.0, and we expect that any matching system that is given two trees that are the same return the maximum similarity score. For SPSM this maximum value is 1.0.

After each alteration operation is applied on the node k, the value of $score_k$ is changed depending on the kind of alteration operation as follows:

Alter word syntactically : for each change in a letter of the word the score is decreased by $(0.5/|word|)$, where $|word|$ is the count of characters of the word. The number of syntactic changes on a word is never higher that $|word|$.

Alter label syntactically : when adding or dropping a word in the node label $score_k = 0.5$

Alter node syntactically : when replacing the label of the node with an unrelated one, or removing the node the $score_k = 0$. Then altering the order of the nodes between the siblings, the $score_k$ remains unchanged.

Semantic alteration : when a word is replaced by a synonym the $score_k$ value remains unchanged; when a word is replaced by an hypernym or hypomyn, the $score_k = 0.5$.

The $score_k$ values were set empirically by trial and error one by one. The rationale behind these particular values is that the similarity should decrease in a

way that reflected the alterations in the trees. The change in similarity also reflects how much a human would say that the original and altered trees were different if the person did not know that one tree was derived from another.

Note that since the probabilities of applying one alteration or another, one node can be affected by more than one alteration. Depending on the combination, the expected score could fall bellow 0, therefore, when this happen, we set the expected score to 0 (zero).

10.3 Results

For each of the 100 original trees, we generated 30 altered trees using the parameters reported in Table 10.1. In order evaluate Structure Preserving Semantic Matching presented in Section 5.3 we have compare it with a set of basic matchers as presented in Section 5.2.2, namely: edit distance, N-gram, prefix, suffix and wordnet relations. We performed three separated tests in order to check how the different alterations affect the matchers: *i)* only syntactic alterations, *ii)* only semantic alterations and *iii)* both alterations combined.

The evaluation consists in matching pairs of original trees oT_i and its altered trees aT_i^j where $i = [1..100]$ and $j = [1..30]$. The experiment was carried out five times in order remove noise in the results. We report here the average results.

Table 10.1: Parameters used for generating and modifying the trees

Parameter	**Syntactic**	**Semantic**	**Combined**
Number of trees	100	100	100
Number of modifications per tree	30	30	30
Average number of nodes per tree	8	8	8
Probability of replace a word of a node for a related one (semantic change)	0.0	0.8	0.8
Probability of making a syntactic change in a word of a node	0.3	0.0	0.3
Probability of changing the order of the words in a node	0.4	0.4	0.4
Probability of deleting or adding a word in a node	0.2	0.2	0.2

Since the tree alterations and the expected similarity scores were known, these provided the ground truth, and hence, the results are available by construction [EIM+07, LSDR07]. This allows for the computation of the matching quality measures such as *Precision*, *Recall* and *F-measure*.

The average efficiency of SPSM in all the test cases is 93ms in a standar laptop with Core Duo CPU-2Ghz and 2GB of ram, running Java 1.4 in Windows Vista, with no other applications running but the matching experiment.

10.3.1 Syntactic Alterations

Figure 10.1 is composed by three plots: *a)* recall vs various cut-off threshold values in [0 1], *b)* precision vs various cut-off threshold values in [0 1], and *c)* F-measure vs various cut-off threshold values in [0 1]. The Figure shows that for the

syntactic alterations, as expected, string-based matchers (edit distance, N-gram, prefix, suffix) outperform the WordNet matcher. Also, edit distance performs as well as SPSM, this is the expected behaviour since, as we presented in Section 5.3, SPSM is based on S-Match, and this particular matcher contains all the string-based matchers mentioned above.

The results from Figure 10.1 show that in absence of semantic variation, and when there are misspellings and other syntactic variations, SPSM can perform as well as the best string-based matcher. However, as we will see in the following subsections, when more semantic information can be derived, SPSM outperforms the string-based matchers. The best performance in terms of F-Measure (which is 0.52) is reached at the threshold of 0.8 by both SPSM and edit distance.

10.3.2 Semantic Alterations

Figure 10.2 is also composed by three plots: *a)* recall vs various cut-off threshold values in [0 1], *b)* precision vs various cut-off threshold values in [0 1], and *c)* F-measure vs various cut-off threshold values in [0 1]. The Figure shows that for the semantic alterations, as expected, the Wordnet-based matcher outperforms the string-based matchers. SPSM outperforms the Wordnet-base matcher, reaching the highest F-measure of 0.73.

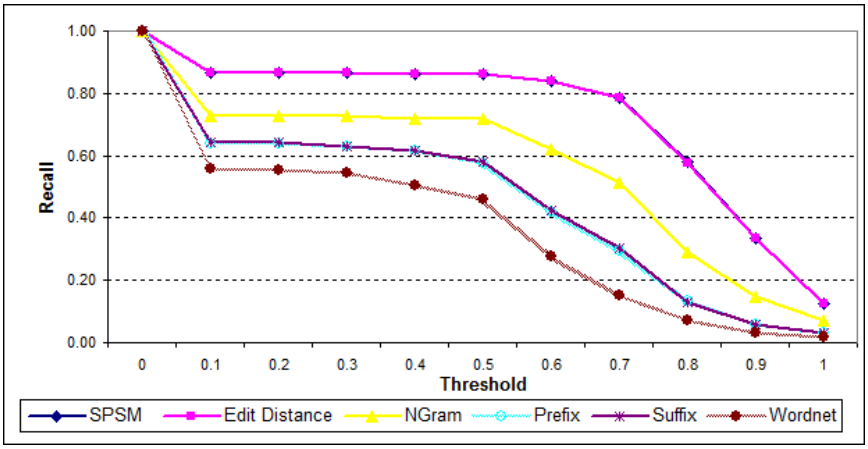
In Figure 10.2 c), we can see how Wordnet and SPSM have similar F-measure up to the cut-off threshold of 0.6, then Wordnet's F-measure start to decrease, while SPSM's F-measure continues to increase. If we consider Figure 10.1 c) (for syntactic variations), we see that SPSM also increases its F-measure until the cut-off threshold of 0.8 with a value of 0.73. If we bear in mind that SPSM uses both syntactic and semantic matchers, we can understand how SPSM outperforms Wordnet, since it also uses syntactic information.

10.3.3 Combined Alteration

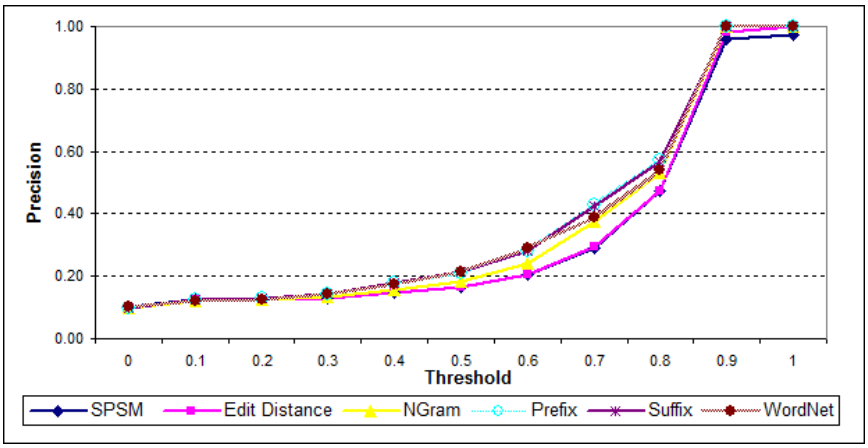
Figure 10.3 is also composed by three plots: *a)* recall vs various cut-off threshold values in [0 1], *b)* precision vs various cut-off threshold values in [0 1], and *c)* F-measure vs various cut-off threshold values in [0 1]. The Figure shows that when combining syntactic and semantic alterations, as expected, SPSM outperforms all the single matchers.

We can see in Figure 10.1 a) that increasing the threshold reduces the correspondences considered to be correct, decreasing recall, the drop starts at 0.6 and becomes more significant between 0.8 and 1. This mean that the highest the cut-off threshold, the least the number of correspondences that are considered to be correct. If we consider Figure 10.1 b), we can see the opposite happening for precision in the same cut-off threshold mentioned before. The more strict we are in considering the correspondences correct, the higher the precision when choosing the correct results.

a)



b)



c)

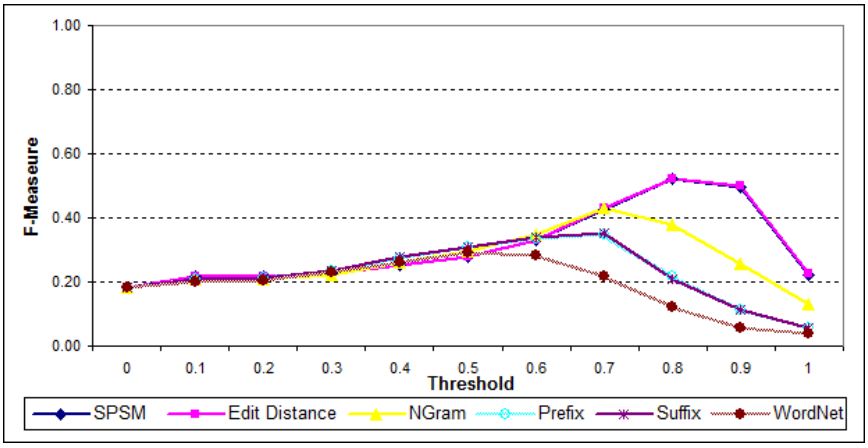
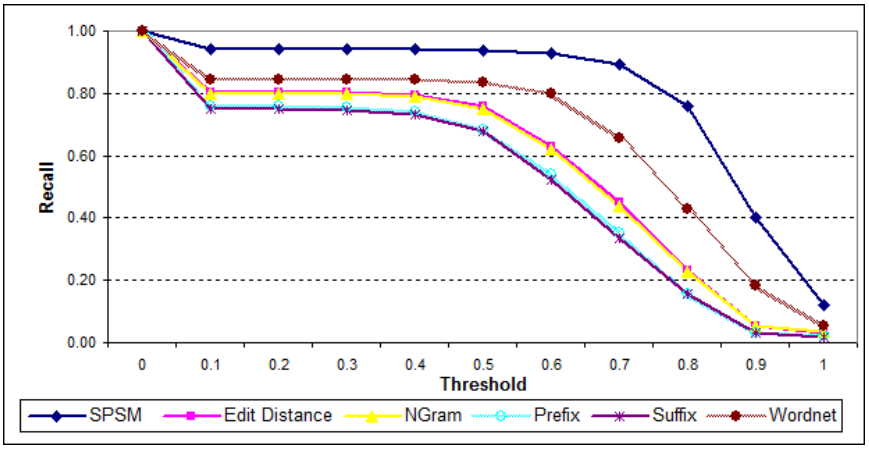
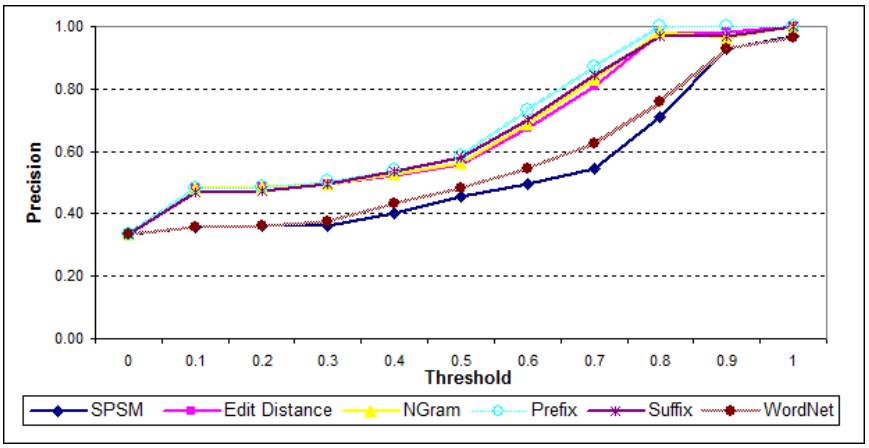


Figure 10.1: Evaluation results for syntactic changes.

a)



b)



c)

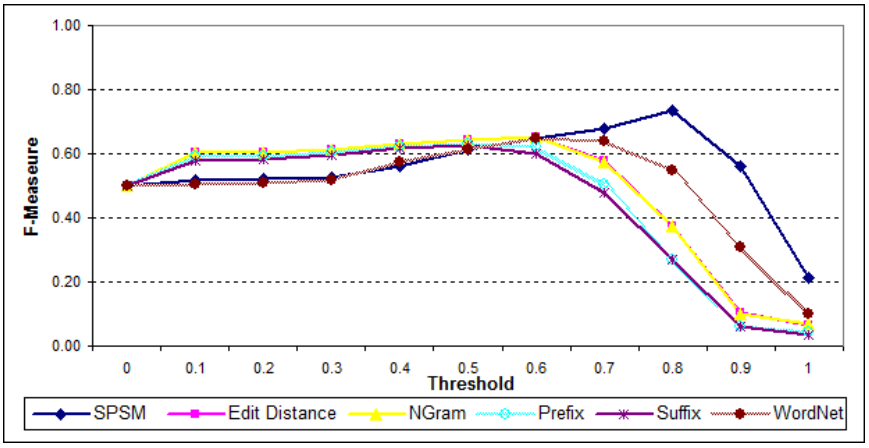


Figure 10.2: Evaluation results for semantic changes.

Figures 10.1 a) and b) show the usual trade-off between precision and recall (the same trade-off can be observed in Figure 10.1 and 10.2). The choice between one and the other normally depends on the application domain. For example, in highly critical scenarios such as health related applications, one would want to be very accurate in the decisions (high precision), without considering how many other possibilities are being missed (low recall), as long as there is at least one match that is near perfect and that can be used.

We can see in Figure 10.3 c), that the best F-measure of 0.47 belongs to SPSM. When faced with syntactic and semantic changes, the second best matcher is edit distance. Up to the 0.6 cut-off threshold, Wordnet is the third best performing matcher, but after that threshold, we see that Wordnet performance decrease and N-gram outperforms Wordnet.

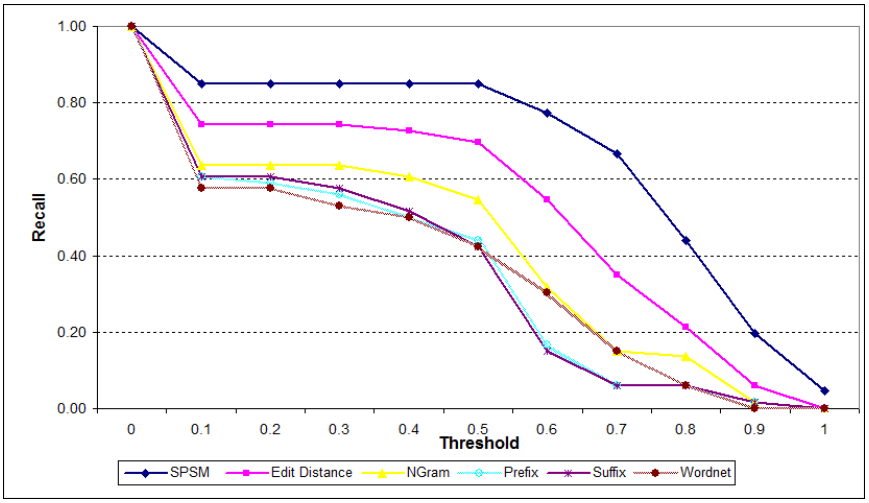
10.4 Summary

In this Chapter we presented an synthetic evaluation of the Structure Preserving Semantic Matching algorithm (SPSM) outlined in Chapter 5. We can see that SPSM is robust to syntactic, semantic and combined alterations with an maximun f-measure of 0.52, 0.73 and 0.47 respectively. We saw that in all the cases SPSM performs at least as well as the individual matchers, but when combining the alterations, which is what will be expected in a real scenario, we see the SPSM outperforms the other single matchers.

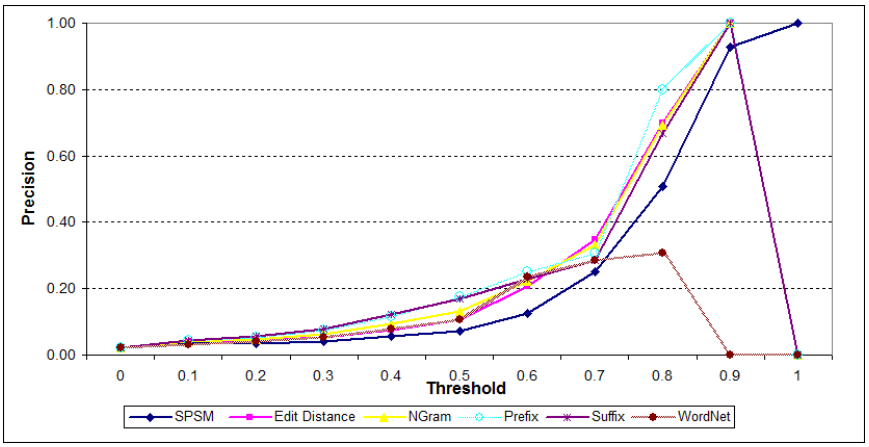
Given the dependency of SPSM on the knowledge available in the background knowledge (the relation between the concepts), the performance is highly dependent on this knowledge. If the knowledge is too general, for example in the case of a knowledge base bootstrapped with Wordnet, SPSM can miss several alignments that a domain expert would find manually. This shows the importance of encoding domain knowledge and shows why the community layer presented in Chapter 4 “The semantic layers” is important.

In order to tackle the issue of the coverage of knowledge we present in Chapter 6 “Background Knowledge Evolution” an algorithm that based on the usesage of terms can update automatically the background knowledge with new terms and concepts to keep up with the dynamic nature of knowledge. In the following Chapter “Evaluation of Background Knowledge Evolution” we show the evaluation of this algorithm.

a)



b)



c)

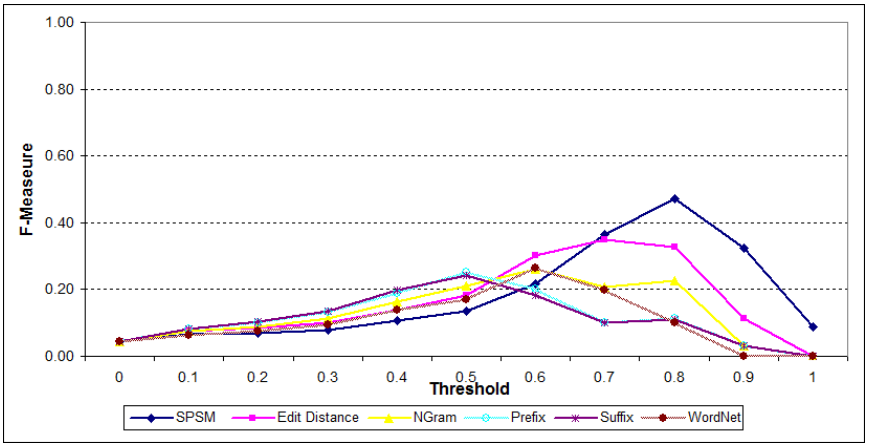


Figure 10.3: Evaluation results for combined changes.

Chapter 11

Evaluation of Background Knowledge Evolution

11.1 Introduction

In Chapter 5 we presented the Semantic Matching algorithm that aims at automatically tackling the semantic heterogeneity problem relying on the Background Knowledge for converting ambiguous labels into formal semantic strings. In Chapter 10 we evaluated Structure Preserving Semantic Matching (SPSM) that, although showing good results, has no perfect recall. This lack of perfect recall is also present in all the state-of-the-art matchers that participated in the Ontology Alignment Evaluation Initiative (OAEI) campaigns.

One of the reasons for the lack of perfect recall is that some matchers rely on background knowledge to derive information. However, if the background knowledge is static, the lack of new terms and concepts of the application domain can become a problem. In order to tackle this issue we proposed in Chapter 6 a sense induction algorithm that, given the usage of words in folksonomies, can infer the senses in which they are being used, recognizing homonyms and synonyms, and add them to the Background Knowledge when missing. To the best of our knowledge, there is no standard mechanism to evaluate such sense induction algorithms. Furthermore, there are no available gold standard datasets that can be used to compare the performance of different approaches.

In order to help address the above mentioned problems we developed *Tags2Con* (an abbreviation for "from tags to concepts"), a tool for creating manually validated gold standards datasets for evaluating semantic annotation systems including folksonomies. The tool includes state-of-the-art algorithms from Natural Language Processing (NLP) and Word Sense Disambiguation (WSD) that automatically suggest senses for the annotations (tags in the folksonomies, see eq. 6.1 on page 57) and provides a manual validation user interface for quickly validating large samples of folksonomies. We also present an evaluation methodology that, together with the developed gold standard dataset, is used to analyze the performance of the

sense induction algorithm.

To the best of our knowledge, the manually validated gold standard dataset is the first openly available dataset that is used for assessing the quality of semantic annotation systems. The dataset can also be used in the fields of NLP and WSD to evaluate tokenization, lemmatization and disambiguation algorithms. Furthermore, while developing the dataset we studied the issue with the coverage of the vocabularies and found that even though the state-of-the-art WSD algorithms report an average performance of 60%, this only applies to the recognized words and concepts. The reality is that, by relying on static linguistic datasets such as Wordnet, the coverage of the words and concepts is only between $\sim$49% and $\sim$71% depending on the application domain. This means that from the end user point of view, in only between $\sim$29% to $\sim$42% of the cases there will be a correct sense suggested by any WSD algorithm relying on these types of linguistic datasets.

The structure of the Chapter is as follows: in Section 11.2 we present the *Tags2Con* tool and a manually validated dataset containing 7 882 links to sense in Wordnet. The dataset is published and available in the Linked Data cloud; In Section 11.3 we define the methodology for evaluating our sense induction algorithm and present preliminary results using state-of-the-art measures; finally, Section 11.4 summarizes the findings of the Chapter.

Acknowledgment: The work presented in this chapter has been carried out with Pierre Andrews and Ilya Zaihrayeu. Parts of this Chapter have been initially reported in [AZP+10] and later published in [APZ11a] and [APZ11b].

11.2 A semantic annotation dataset

Since the work on the extraction of formal semantics from folksonomies became an active topic of research in Semantic Web and related communities, the research community realised that the lack of golden standards and benchmarks significantly hinders the progress in this area: *"The current lack of transparency about available social tagging datasets holds back research progress and leads to a number of problems including issues of efficiency, reproducibility, comparability and ultimately validity of research"* [KS10]; and in particular about finding semantics in folksonomies we have noticed the lack of testbeds and standard evaluation metrics that allow proper comparisons of the different research works [GSSAC09, KS10, GSCAGP10].

As reported in [KS10], some datasets are available for different folksonomies, however, none of them is yet annotated with links to an ontology disambiguating the tags' meaning (their semantic). For instance, in the Delicious folksonomy[1], if a bookmark is tagged with "javaisland", to know the real semantic of such tag, we need to split the tag in the two tokens "java" and "island" and then connect each token to its corresponding unambiguous sense: "java – an island in Indonesia

[1] `http://del.icio.us`

to the south of Borneo" and "island – a land mass that is surrounded by water" (disambiguation, as illustrated in Figure 11.1).

In fact, if we look at some of the latest publications in the field, we can see that such call for evaluation datasets is well grounded. For instance, in [LDZ09], the authors try to extract ontological structures from folksonomies but only report a "subjective evaluation" where they study manually a generated ontology with four branches. To perform a real quantitative evaluation of their algorithm, the authors would need a folksonomy of which the tags are already linked to an existing ontology.

In line with what has been requested in other publications [GSSAC09, KS10, GSCAGP10], we thus believe that a new type of dataset is needed that provides a real golden standard for the links between the folksonomy tags and their semantics. To guarantee the validity of such a golden standard, the tag cleaning and disambiguation has to be performed manually by a team of annotators. In the following sections we describe a tool for creating such gold standard dataset, a initial (extensible) dataset, and some results obtained on a golden standard that we built on a sample of Delicious.

11.2.1 A tool for Creating Golden Standards of Semantic Annotation Systems

In order to help address the problem mentioned in the previous section and to create a golden standard dataset to be used for the evaluation, we developed a tool for creating golden standards for semantic annotation systems, called *Tags2Con* (an abbreviation for "from tags to concepts"). The aim of the tool is to provide a manual validation interface for quickly annotating large samples of folksonomies to create the required golden standards.

11.2.1.1 Automatic preprocessing

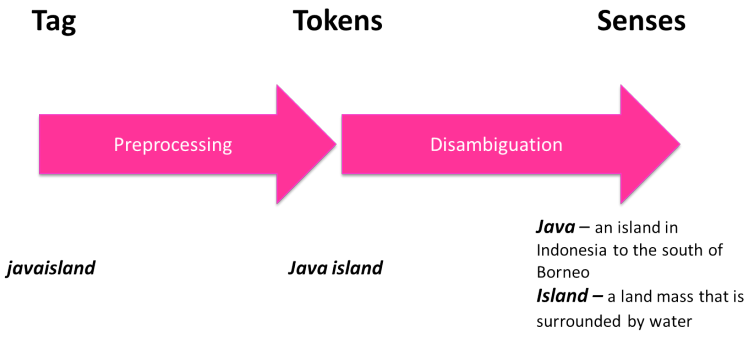


Figure 11.1: Semantic Folksonomy Dataset Creation Process.

The tool provides a set of utilities and algorithms for the semi-automatic conversion of free-text annotations (see eq. 6.1 in page 57) into semantic annotations (see eq. 6.2 in page 58), i.e., ($t \rightarrow string_{sem}$). It automatically performs the preprocesssing steps outlined in Figure 11.1, these are[2]:

- **Tokenization**: splittings the tags into the component tokens (words). This step generates different splits for the same tag, for instance, the tag "javaisland" can be split into {"java", "island"} or {"java", "is", "land"}. The output of this step is ranked to present the most plausible split to the annotator first. The ranking prefers proposals with a fewer number of splits and with the maximum number of tokens linked to the controlled vocabulary.

- **Lemmatization**: in order to reduce different forms of the word into a single form (that can later be found in the linguistic part of the Background Knowledge).

- **Disambiguation**: tries to select the correct sense of the words recognized by the previous steps. The algorithm we use is an extension of the one reported in [ZSG+07] based on the idea that collocated tags provide context for disambiguation (as generalised in the survey by Garcia-Silva [GSCAGP10]). In our approach, given a token within a tag split, we consider three levels of context:

  1. the other tokens in the tag split provide the first level of context,
  2. the tokens in the tag splits for the other tags used for the annotation of the same resource by the *same* user provide the second level,
  3. the tokens in the tag splits for the tags used for the annotation of the same resource by *other* users, provide the third level of context.

  The sense with the highest score is then proposed to the validator as the suggested meaning of the token (see [AZP+10] from more details).

Once the tool has computed the possible splits and senses for the tags, the validator is asked to validate the split and senses, or to choose the correct ones.

11.2.1.2 Manual validation

The *Tags2Con* tool main task is to ask validators to provide the true disambiguation of tags. The process of validation includes the selection of the correct tag split, and for all the tokens (the words) in the tag split, the selection of the correct sense. Figure 11.2 shows the user interface developed for the validation task. In order to be certain that the selected sense is the correct one, we control annotator agreement by having all the tags validated by at least two different validators.

The manual validation process is summarized as follows:

[2] See Section 5.2.1 for more details on tokenization and lemmatization

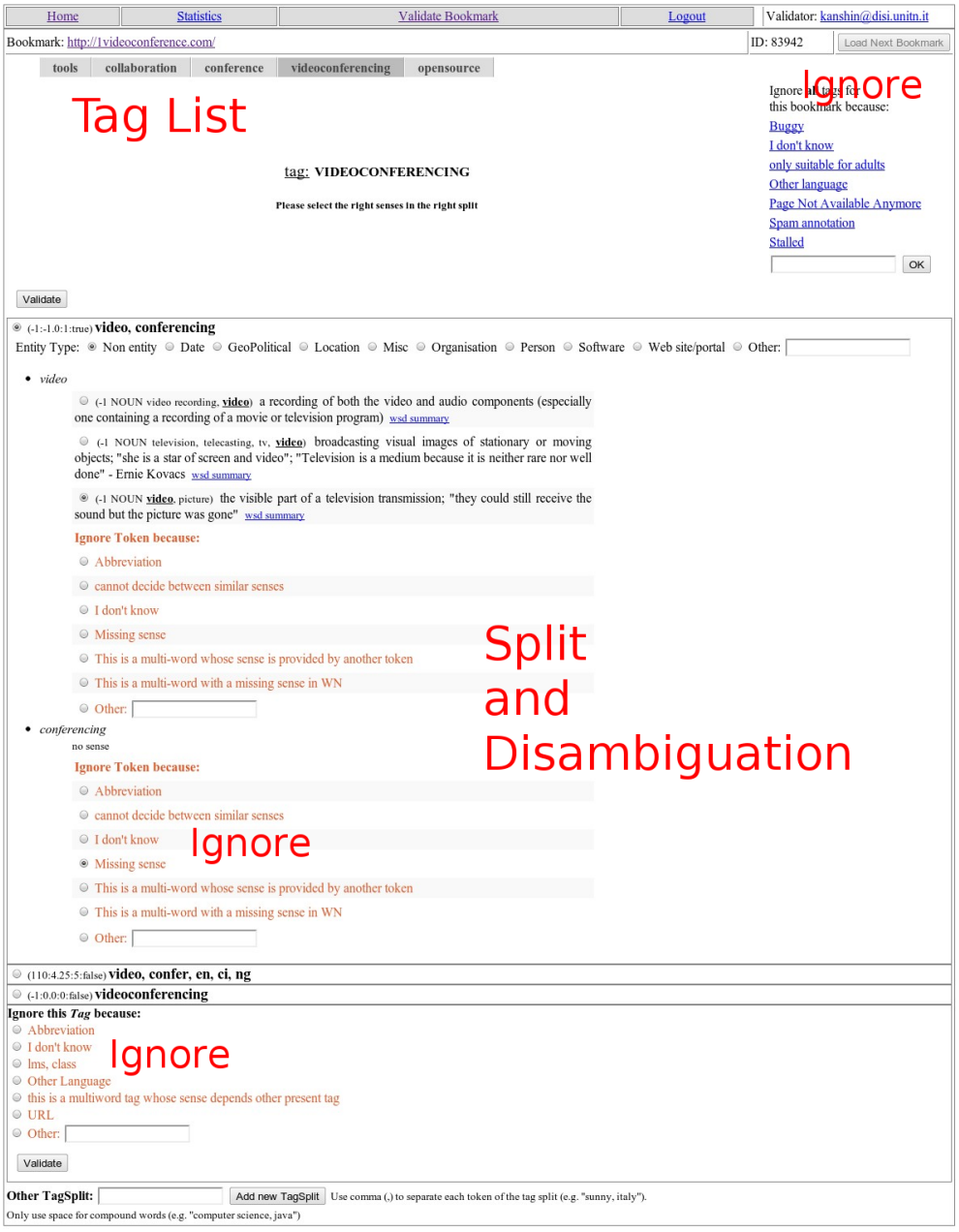


Figure 11.2: Annotation Page for the Manual Tag Disambiguation

1. A $\langle resource, user \rangle$ pair is selected automatically to ensure that all tags for the $\langle r, u \rangle$ pairs are validated twice to reach a certain confidence in the validation process. Once the resource (the web page) is selected, it is visualised in another window of the browser so that the validator can check the Web page and therefore understand the context and the possible intended meanings of the tags. The validator has the option of ignoring the given resource at the top right part of the screen if, for example, the Web site is not available anymore or it is in a foreign language.

2. the validator selects a particular tag for the given $\langle r, u \rangle$ pair from the Tag List on the top left part of the user interface. In the example of Figure 11.2 we can see at the top that the tag "videoconferencing" is selected.

3. the validator selects the correct tag split from the existing list. In the example of Figure 11.2 the selected split for the tag "videoconferencing" is {"video", "conferencing"} instead of {"video", "confer", "en", "ci", "ng"} or {"videoconferencing"}. The validator is also given the option of ignoring the selected tag if no meaningful tag split or sense is suggested. The validator also is presented with the option of suggesting the correct tag split (the "Add new TagSplit" button at the bottom left part of the user interface in Figure 11.2);

4. the validator selects the correct sense for each of the tokens (words) in the tag. For example, for the word "video", WordNet shows three senses as shown in Figure 11.2. The user can also choose to ignore the token (word) if the correct sense is not present in the list, indicating the reason; for example because the correct sense is missing from the controlled vocabulary (this functionality will later allow us to compute the coverage of Wordnet);

5. the validator submits the results and validates the next tag, until all the tags have been validated for the $\langle r, u \rangle$ pair, in which case the validation page goes to the next $\langle r, u \rangle$ pair.

Selecting the right tag split and the right disambiguation for each token in such split is a tedious task for the human annotators and *Tags2Con* tries to make this task as straightforward as possible. It also provides some supporting tools to simplify the work of the validators and streamline the validation process. With this platform, a team of four validators annotated a sample of 7,323 tags from $\sim$1,500 $\langle r, u \rangle$ pairs from a Delicious crawl in less than two weeks working on their spare time (See Section 11.2.2). To enable such streamlined annotation, the automatic computation of the most probable splits and senses prove to be of key importance.

11.2.2 Semantifying a Delicious Dataset

Given the sense induction algorithms discussed in Chapter 6 "Background Knowledge Evolution" and the evaluation methodology in Section 11.3.1, we need a

golden standard dataset that complies with semantic annotation defined in eq. 6.2 in page 58, i.e., we need to map the tags t to concepts via semantic strings. In the current state of the art of the semi-automatic knowledge building field, folksonomies such as Delicious[3] or Flickr[4] are used as data providers to study the automatic construction of ontologies and the evolution of uncontrolled vocabularies. We select subset of a Delicious dump and disambiguate the tags using the *Tags2Con* platform described in the previous Section.

11.2.2.1 Delicious Sample

We obtained the initial crawled data from Delicious between December 2007 and April 2008 from the authors of [WZB08]. After some initial cleaning (eliminate invalid URLs, tags with only numbers and non-ASCII characters) the dataset contains *5 431 804* unique tags (where the uniqueness criteria is the exact string match) of *947 729* anonimized users, over *45 600 619* unique URLs on *8 213 547* different website domains. This dataset can be considered to follow free-text annotation model presented in eq. 6.1 in page 57, where the resource r is the URL being annotated. There is a total of *401 970 328* free-text tag annotations in the crawled cleaned dataset. Note that this is the largest crawled dataset of Delicious currently available.

The golden standard dataset we have built includes annotations from users which have less than 1 000 tags who have created at least ten different tags in five different website domains. This upper bound was decided considering that Delicious is subject to spam annotations, and users with more than one thousand tags could potentially be spammers or machine generated tags as the original authors of the crawled data assumed [WZB08]. Furthermore, only $\langle r, u \rangle$ pairs that have at least three tags (to provide diversity in the golden standard), no more than ten tags (to avoid timely manual validation) were selected. Only URLs that have been used by at least twenty users are considered in the golden standard in order to provide enough overlap between users in order to allow the semantic convergence and evolution algorithms to be evaluated. After retrieving all the $\langle r, u \rangle$ pairs that comply with the previously mentioned constraints, we selected 1 681 pairs in two batches with the following methods:

1. **Random**: 500 pairs were selected purely at random,
2. **Domains**: 1 181 pairs were selected randomly at equal distribution in the pairs that overlaped with one of the following DMOZ[5] topics: /Home/Cooking, /Recreation/Travel or /Reference/Education. This addition was made in order to include in the dataset more tags and resources from general domains as the first random batch yielded a dataset biased towards Computer Science.

Table 11.1 summarizes the characteristics of the resulting subset of the dataset.

[3] `http://delicious.com`
[4] `http://flickr.com`
[5] `http://www.dmoz.org/`

Item	Count
$\langle user, resource \rangle$ pairs	1 681
unique tags	1 569
unique website domains	603
unique URLs of resources	739
unique users	1 397
manually validated tags instances	7 323
manually validated tokens instances	7 882
unique tokens with senses in WordNet	644
unique senses linked to WordNet	689

Table 11.1: Statistics of the tags2con-delicious gold standard dataset

The resulting *Tags2Con* gold standard dataset is publicly available[6] for reuse following the RDF format depicted in Figure 11.3. The dataset is also part of the Linked Open Data(LOD) cloud[7].

11.2.2.2 Results

In the following paragraphs we outline the results for:

- the automatic preprocesssing: tokenization and WSD,
- the manual validation process,
- the type of vocabulary used by the user of Delicious,
- the coverage of the Wordnet vocabulary for the two batches,
- some statistics on the whole dataset (not only the manually validated part).

A more detailed analysis can be found in [AZP+10].

Tokenization The accuracy of the preprocessing step (see Section 11.2.1.1) in this validation task reached 97.24%. Figure 11.4 shows an analysis of the accuracy of the algorithm for different numbers of possible splits. The Y axis corresponds to the distribution of tags per number of possible splits proposed to the validator, the top box is the amount of wrong split while the bottom one represents the amount of accurate splits that were ranked top by the preprocessing algorithm. The figure should be read as follows: the number of tags with two possible splits is ∼35% and

[6] `http://disi.unitn.it/knowdive/dataset/delicious/`

[7] Linked Open Data cloud diagram, September 19th 2011, by Richard Cyganiak and Anja Jentzsch. `http://lod-cloud.net/`

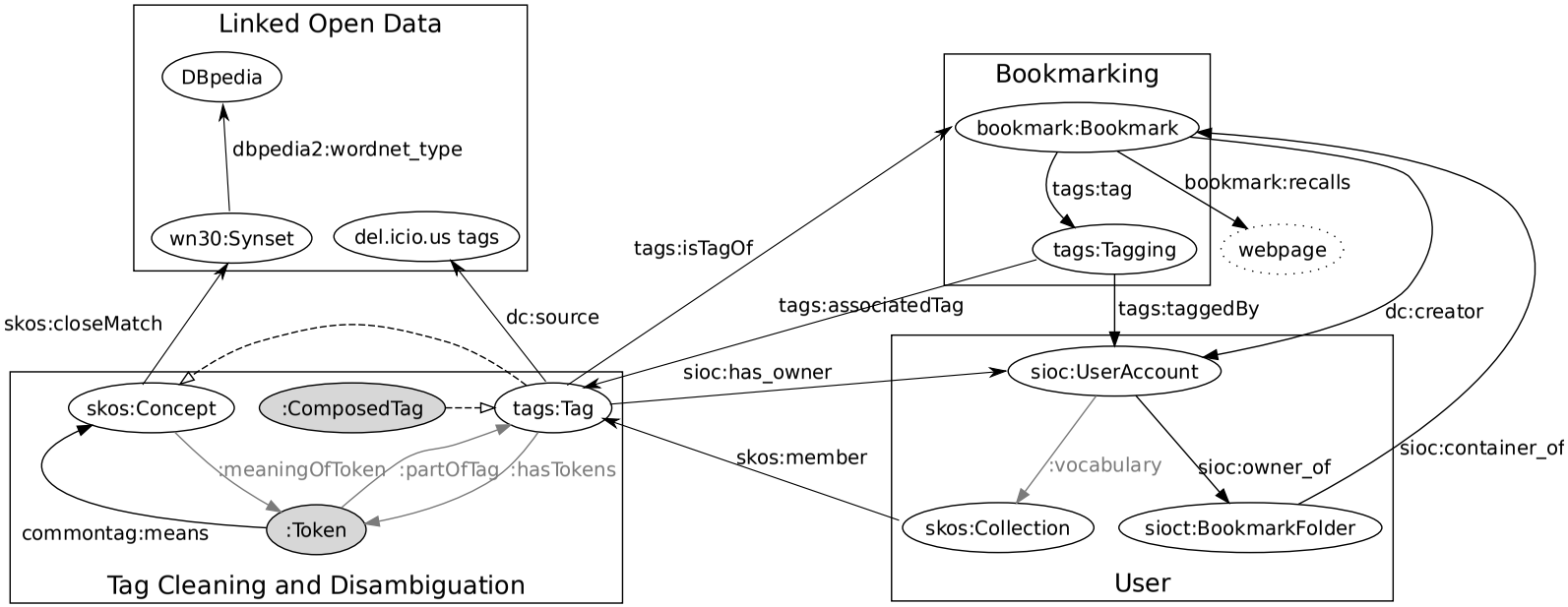


Figure 11.3: RDF Mapping for the Semantified Delicious dataset

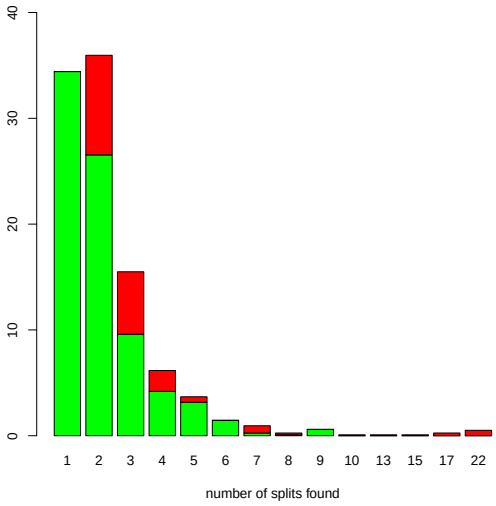


Figure 11.4: Accuracy of the Preprocessing algorithm by the number of possible splits proposed to the validator.

the accuracy of the algorithm for these cases is $\sim$80% (see the second bar from the left).

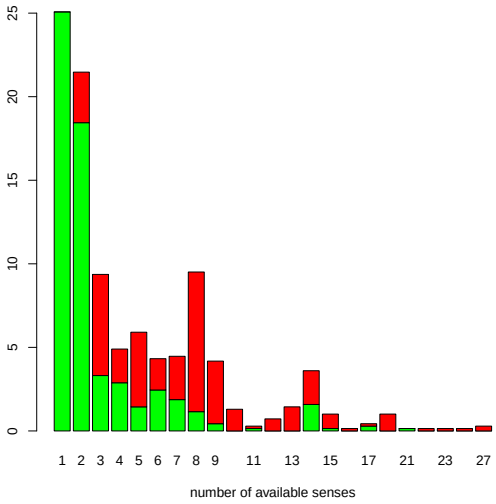


Figure 11.5: Accuracy of the WSD algorithm by the level of polysemy.

Word Sense Disambiguation The average polysemy of the tag tokens in the validated dataset was 4.68, i.e., each tag token had 4.68 possible senses on average. The implemented WSD algorithm reached an average accuracy of 59.37%. Figure 11.5 shows an analysis of the accuracy of the algorithm for different levels of polysemy. The Y axis corresponds to the distribution of tokens per polysemy, the top box is the amount of wrong disambiguation while the bottom one represents the amount of accurate disambiguations that were ranked top by the WSD algorithm. The Figure should be read as follows: the number of cases with two possible senses in the controlled vocabulary is $\sim$22% and the accuracy of the algorithm for these cases is $\sim$90% (see the second bar from the left).

It is worth noting on Figure 11.5 that the WSD algorithm has an accuracy lower than 50% for the tokens with many available senses, however, the majority of the tokens have two senses, and in these cases the WSD algorithm reached an accuracy of $\sim$90%. We can see also in the Figure that relying on a general purpose lexical database such as Wordnet, which tries to include all and every possible sense for the recognized words, makes it very difficult to manually decide between the available senses, which in some cases were up to 27.

Manual Validation In order to guarantee the correctness of the assignment of tag splits and tag token senses, two different validators validated each $\langle r, u \rangle$ pair. As the result, we obtained 7 323 tag validations; 91% of $\langle r, u \rangle$ pairs in the dataset were marked as valid, i.e., the user selected a tag split and disambiguation; 8,9% were marked as invalid because, for instance, the Web page was not available anymore, it was in another language or the page was only suitable for adults.

Among the $\langle r, u \rangle$ pairs marked as valid, 84.69% of the tags could be validated by the validators while the other tags were ignored, mostly because the validator did not know what the tag meant, the tag was in another language, or consisted to be part of a multiword tag[8].

The "agreement without chance correction" [AP08] between users in the task of disambiguation of tokens is 0.76. As mentioned in [AP08], there is not yet any accepted best measure for the agreement in the task of sense annotation and thus we currently report only the raw agreement. It is intuitive to think that the agreement will fall when there are more available senses for one token as the annotators will have more chances to choose a different sense. This could also happen because, as we show in Figure 11.6, sometimes the annotators cannot decide between too fine-grained senses in the controlled vocabulary.

Use of Nouns, Verbs and Adjectives A previous study of Delicious [DG09] points out that the users of folksonomies tend to use mainly nouns as descriptors of the URLs. In the *Tags2Con* dataset nouns represent 88.18% of the validated

[8]this ignore option was used when the meaning of one tag was dependent on another different tag, for example, when two tags as in "George" and "Bush" were applied – which derives from the issue of using space as separator in Delicious

tags, while adjectives represent 9.37% and verbs only 2.45%. We have not found adverbs in the validated dataset.

Coverage of the Background Knowledge While disambiguating the tags to a sense in WordNet[9] the manual annotators could also decide that no sense provided by the vocabulary was adequate to express the sense meant by the user, even if the word was recognized. For example, the tag "ajax" was found in the dataset usually referred to the ajax technology used in Web applications[10]. However, the only sense present in WordNet for this tag is "a mythical Greek hero".

As shown in Figure 11.6 a), in the randomly selected batch the validators were able to find the correct sense in 49% of the cases. However, in 36% of the cases the validators said that even though the tag was properly split and the words were correct, the sense was missing in the list provided by the tool (recall the "ajax" example). In 6% of the cases, the users were not able to distinguish between the senses that were provided to them and choose the correct one, normally because there was no much difference between the provided senses (for example, "bank" has ten different senses as a noun, and eight different sense as verb). For these and other reasons, less than half of the vocabulary used by the users could be mapped to an existing sense in the "Random" batch.

When first validating the tags with the randomly selected batch, we already realize that the coverage of the vocabulary was low. After investigating the properties of Delicious, it was obvious that any random selection would lead to the same result, as most of the tags are related to computer science.

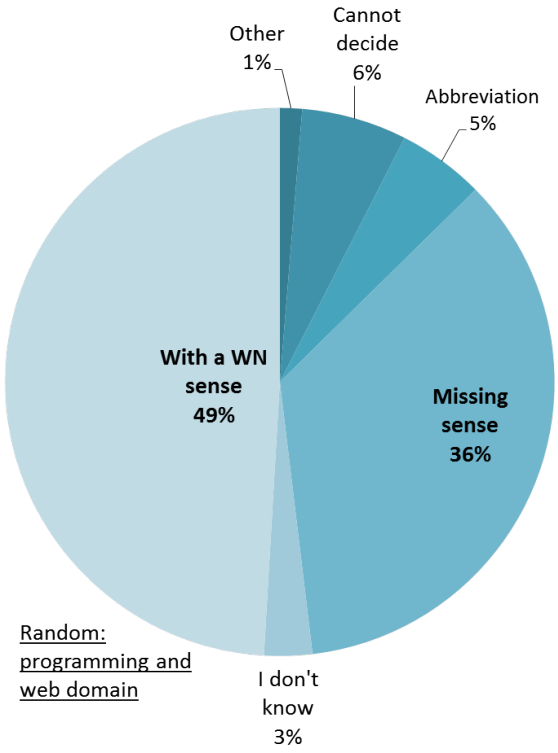
In order to analyze the coverage of the vocabulary in other domains, we created the second batch of annotations including resources (URLs) from the domains of Cooking, Travel and Education. Figure 11.6 b) shows the result of the coverage of the language in this case. In 71% of the cases the validators were able to find the correct sense. This shows a clear increase (22%) in the coverage of the language with respect to Computer Science (the first batch). In 15% of the cases the validators said that even though the tag was properly split and the words was correct, the sense was missing. This represents less than half of the missing senses with respect to Computer Science. The high complexity for distinguishing the proper senses in a long list of senses remained similar, 5% (as opposed to 6%).

These findings with respect to the coverage of the vocabulary are important as they show the inadequacy of fully automatic folksonomy processing systems based on fixed controlled vocabularies such as WordNet. For instance, if we consider the issue of Word Sense Disambiguation, the state-of-the-art tools often perform with an accuracy of 60%. However, given the fact that only around half of the terms can be found in a vocabulary for the domain of Computer Science, from the end user perspective, it means that in only $\sim$29.4% of the times the user will be suggested the right sense. For other domains (Cooking, Travel and Education) the user is

[9] from where we initially populated our Background Knowledge.

[10] `http://en.wikipedia.org/wiki/Ajax_(programming)`

a)



b)

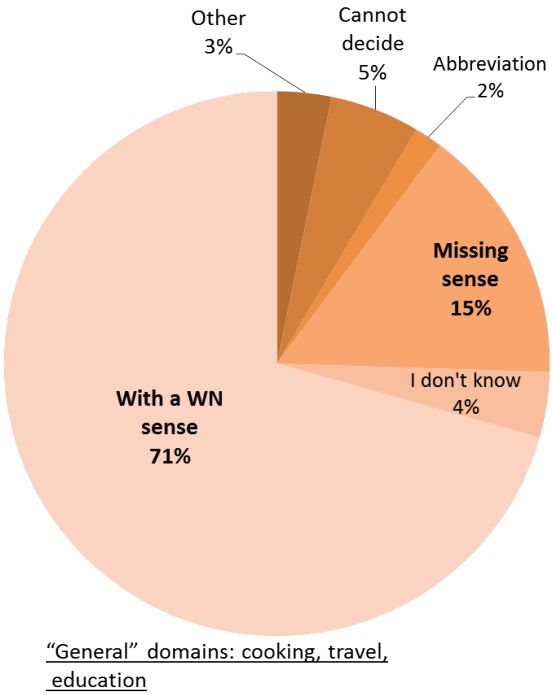


Figure 11.6: Distribution of validated tokens by type of validated dataset.

suggested to correct term in ∼42.6% of the times.

Considerations on the Dataset Uncontrolled Vocabulary In the following paragraphs, we present some analysis that we performed on the whole dataset of 45 600 619 URLs, with all the users and tags available.

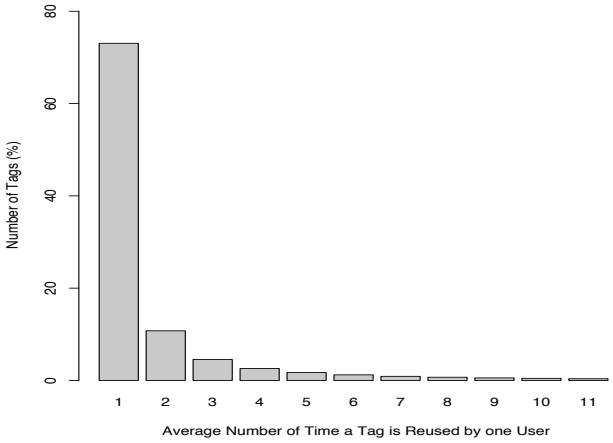


Figure 11.7: Number of times a tag is reused by the same User on all the Bookmarks

In Figure 11.7, we analyze tag reuse by users on Delicious. While an often used assumptions in folksonomy studies is that we can learn a lot from tag collocations on different resources, we can see that users do not often reuse the same tag more than once. In fact, from our analysis, in 73% of the cases, a tag is used only once on the whole set of bookmarks by a single user. This means that in a majority of the cases, a tag will not be found located on different resources, at least not by the same user. Only in ∼10% of the cases the user applies the same tag on two different resources. Only in 7.3% of the cases a tag is reused on more than three resources.

This finding might impair the ontology learning algorithms [GSCAGP10] that are based on the measure of collocation of tags, as we will see in the results of Section 11.3.2. Furthermore, in [AZP+10] we show that in only 18% of the cases two users use the same tag in the same resource and in 9.3% of the cases more than three users agreed on at least one tag.

11.3 Sense induction evaluation

11.3.1 Methodology

In this section we present the evaluation methodology for the sense induction algorithm presented in Chapter 6 "'Background Knowledge Evolution". As pointed out in the previously mentioned Chapter, the Natural Language Processing (NLP) field has already tackled the issue of concepts extraction from text and has considered different evaluation methodologies for this task. [AM02b] describes the evaluation problem as follows:

> Let us suppose that we have a set of unknown concepts that appear in the test set and are relevant for a specific domain: $U = \{u_1, u_2, \ldots, u_n\}$. A human annotator has specified, for each unknown concept u_j , its maximally specific generalisations from the ontology: $G_j = \{g_{j,1}, g_{j,2}, \ldots, g_{j,m_j}\}$.
>
> Let us suppose that an algorithm decided that the unknown concepts that are relevant are $C = \{c_1, c_2, \ldots, c_l\}$. For each C_i, the algorithm has to provide a list of maximally specific generalisations from the ontology: $H_i = \{h_{i,1}, h_{i,2}, \ldots, h_{i,p_i}\}$ (See Figure 11.8).

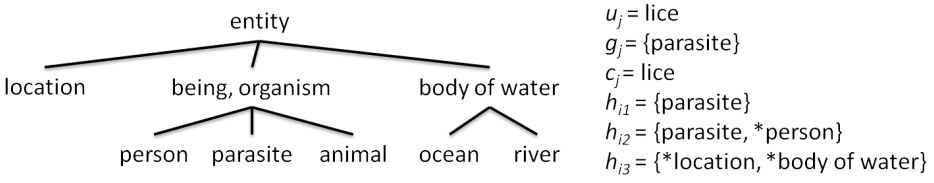


Figure 11.8: Example of taxonomy, an unknown relevant concept u_j, its correct generalisations g_j and the generalisations proposed by three hypothetical algorithms h_{ik}, adapted from [AM02b]

From this definition, a number of evaluation metrics can be computed:

Accuracy: The amount of correctly identified maximally specific generalisations.

Parsinomy: The amount of concepts for which a correct set of generalisations is identified.

Recall: The amount of concepts that were correctly detected and to which at least one relevant maximally specific generalisation was found.

Precision: The ratio of concepts that were correctly attached to their maximally specific generalisations to the total of concepts identified.

Production: The amount of proposed maximally specific generalisations per concept.

Learning Accuracy: The distance, in the concept hierarchy, from the concept proposed placement to its true placement (from [HS98]).

As can be seen from these proposed measures, a gold standard needs to be available that provides the "maximally specific generalisations" (G_j) for each concept (U). [AM02b] use a dataset of textual documents that is manually annotated for this purpose. However, we need to evaluate the algorithm within a folksonomy and thus we use the dataset described in Section 11.2 (the *Tags2Con* dataset) as it provides a manually validated disambiguation for each tag in a subset of the Delicious folksonomy.

The measures listed above can be computed on this dataset by applying a leave one out cross validation approach to the evaluation. That is, we iterate through all tag annotations already linked to a concept in the gold standard; we "forget" the senses of one tag at a time and apply the algorithm on this tag; we then compare the detected senses and their new place in the taxonomy for this tag to the actual sense that the gold standard defines.

While this is a possible evaluation procedure to evaluate the final output of the whole algorithm, the current dataset is not in a form that allows for the evaluation of the intermediate results. In particular, to optimise the λ and m parameters of the clustering algorithm used in the first and second step of the algorithm (see Section 6.4.2 on page 62), we have to be able to evaluate independently the accuracy of each stage of the algorithm. In the same way, we need to be able to evaluate the distance measures used and compare different approaches. For this, we need a clustering gold standard, that provides the "true cluster" (class) of each $\langle r, u \rangle$ pairs in the dataset so that we compare the found clusters to this gold standard results. In the following paragraphs we discuss a strategy to generate such a clustering gold standard.

Let us assume that we have a manually validated dataset of annotations with their corresponding concepts. We assume this manually validated dataset to be the gold standard GS, where the set of unknown concepts $U = \{u_1, u_2, \ldots u_n\}$ to be clustered are the terms used in the annotation; for each unknown concept u_j, the maximally specified generalization G_j from the ontology is the parent concept in the Background Knowledge[11]. In this case, the set of unknown concepts that the algorithm decides to be relevant (C) are the same as the manually validated ones in U, as this set will be generated by the procedure described bellow. Finally, for each $u_i \in U$ (or c_i, as they are the same) the clustering algorithm produces a set of results $H_i = \{h_{i,1}, h_{i,2}, \ldots, h_{i,p_i}\}$ to be compared with the expected results in G_j.

In our case, GS is the manually annotated *Tags2Con* dataset described in Section 11.2. The set of unknown concepts U will be the tokens contained in t in the annotations. The set of maximally specified generalization G_j is the set of parent concepts of the manually disambiguated terms u^i, considering the hypernym relation in WordNet. The construction of the gold standard consists in aligning the set

[11] For example, using the *is-a* relation available in Wordnet.

of concepts U for which we have the resulting clusters G_j and the set of concepts C_i to be given as input to the clustering algorithm, as described below.

When building the gold standard (GS^j) we automatically generate the set of unknown concepts (U and C_i) to be clustered, their classes, and the maximally specified generalization G_j. In order to do so, we perform the following steps:

1. we define G_j to be a concept in our Background Knowledge for which there is more than one hyponym that has more than one manually validated term in the annotation. In the example in Figure 11.9 a), the concept G_1 ="being, organism" has two hyponyms ("person" and "parasite") that contain more than one annotation attached to them, also the concept G_2 ="body of water" has two hyponyms ("ocean" and "river") that have more than one annotation attached to it. Each of the complying concepts ("being, organism" and "body of water") will generate a set of clusters for the gold standard datasets $GS^1_{being,organism}$ and $GS^2_{bodyofwater}$.

2. we "forget" momentarily that each of the hyponyms of G_j exist. Since for each of these children C_i we know their corresponding annotations, we create a class for each deleted concept, and define the boundary of the GS^k clusters to these particular classes. In our example in Figure 11.9 b), we can see that two gold standards have been created: GS^1 for "being, organism" and GS^2 for "body of water", each of them containing two clusters (one for each deleted concept).

3. Starting from the leaves, we recursively repeat the process by further "forgetting" concepts higher in the hierarchy and thus creating more gold standard sets of increasing difficulty as we go up in the hierarchy, the more classes will be created in GS^k. In our example in Figure 11.9 c), we further "forget" the concepts "being, organism" and "body of water" and create another gold standard GS^3 for "entity", creating four clusters for each class.

If we apply the above mentioned process on the dataset depicted in Figure 11.9 a) we obtain three GS datasets as shown in Table 11.2. When using the *Tags2Con* dataset, we can create 4 427 of these gold standard annotations representing manually validated user-bookmark-tagsense triplets, from these we build 857 different GS at various depth of the WordNet *is-a* graph. Section 11.3.2 discusses the results of an evaluation ran with this methodology and dataset.

The purpose of each gold standard GS^k is twofold:

1. Evaluate step one of the sense induction algorithm presented in Section 6.4.2, where the input is a set of free-text tags C and the output is a set of clusters of similar tags that represents a new concept. In our example in Figure 11.9 a), the clustering algorithm can be evaluated on each of the gold standards GS^k. Then, to compute the accuracy of the clustering, we compare the produced results H_i with the classes of the gold standard with standard cluster evaluation metric such as Purity, Accuracy and Precision/Recall [AGAV09].

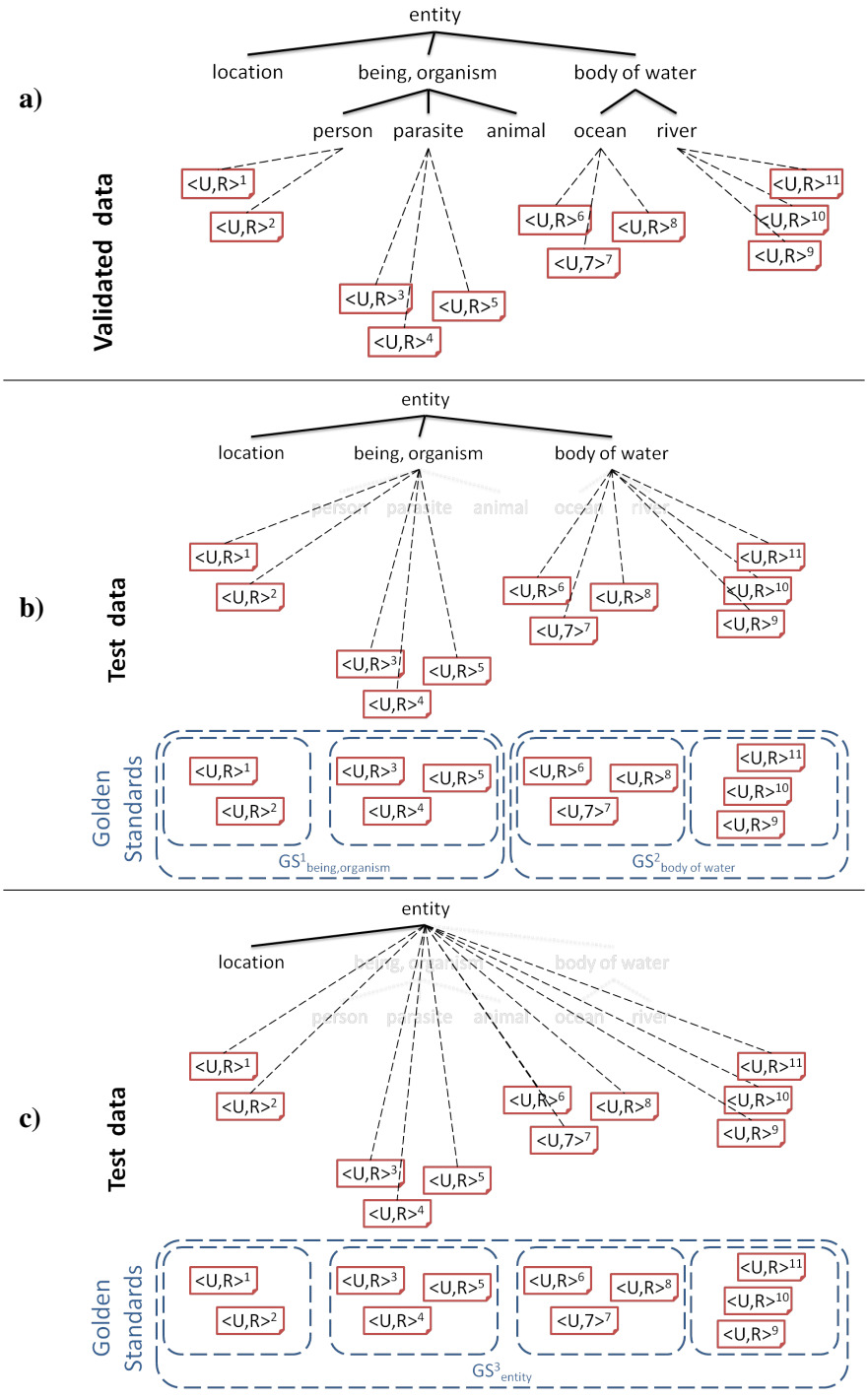


Figure 11.9: Process of constructing an evaluation dataset: **a)** The validated data for which the concepts is known; **b)** Creation of two validation clusters by deleting the children of *being, organism* and *body of water*; **c)** Creation of a third validation cluster by further deleting *being, organism* and *body of water*.

GS^k	C, U	G_j	Clusters and new concepts
$GS^1_{being,organism}$	$C = U =$ {person, parasite}	$G_1 =$ {"being, organism"}	person $= \langle U, R\rangle^1, \langle U, R\rangle^2$ parasite $= \langle U, R\rangle^3, \langle U, R\rangle^4, \langle U, R\rangle^5$
$GS^2_{bodyofwater}$	$C = U =$ {ocean, river}	$G_2 =$ {"body of water"}	ocean $= \langle U, R\rangle^6, \langle U, R\rangle^7, \langle U, R\rangle^8$ river $= \langle U, R\rangle^9, \langle U, R\rangle^10, \langle U, R\rangle^{11}$
GS^3_{entity}	$C = U =$ {person, parasite, ocean, river}	$G_3 =$ {"entity"}	person $= \langle U, R\rangle^1, \langle U, R\rangle^2$ parasite $= \langle U, R\rangle^3, \langle U, R\rangle^4, \langle U, R\rangle^5$ river $= \langle U, R\rangle^9, \langle U, R\rangle^10, \langle U, R\rangle^{11}$ river $= \langle U, R\rangle^9, \langle U, R\rangle^10, \langle U, R\rangle^{11}$

Table 11.2: Resulting gold standards GS^k for the evaluation of the sense induction algorithm

2. Considering that we know the parent concept G_j for each gold standard GS^k, we also evaluate step three of the sense induction algorithm where for each cluster produced, a new concept also has to be added to the KOS as more specific than an existing concept. The generalisations discovered by the algorithm (H_i) is compared to the one given in the gold standard (G_j). In our example in Figure 11.9 a), if we pass GS^1 to the algorithm, it should create concepts for "person" and "parasite", and put them as hyponyms of "being, organism".

11.3.2 Preliminary results

In the previous sections we presented a new evaluation methodology to evaluate each step of the sense induction algorithm presented in Section 6.4. We also presented a gold standard dataset (see Section 11.2.2) that was manually validated to be used with the evaluation methodology

In this section we extend the work carried out in strong collaboration and guidance of Andrews (outlined in [APZ11b]) and report the preliminary results from the two first steps of the algorithm that involve clustering and for which the distance measures introduced in Section 6.5. In particular as these are the steps that are present in the state of the art.

In this section we report precision/recall as an external measure of the clustering quality. We also report F_1 that is the harmonic mean of the two values. However, note that here precision/recall measures the quality of the clustering decision process for each pair of elements to cluster and not the quality of a classification of instances in classes (see [CDMS08] for details of these measures).

The normal state of the art distances used when studying folksonomies were introduced in Section 6.5 and are based on counting collocation of links between elements of the folksonomy graph as defined by Equation (6.1). In particular, we initially studied three collocation measures to compare annotations:

A. *Tag Collocation* The number of collocated tags on the resources of two anno-

tations.

B. ***User Collocation*** The number of users that annotated the resources of two annotations.

C. ***Resource Collocation*** The number of resources two users share.

Distances	Step	Precision (%)	Recall (%)	F_1-Measure (%)
A. Tag	One	53.4	**73.2**	57.2
	Two	0.03	*68.2*	0.06
B. User	One	**58.2**	63.6	56.8
	Two	0.03	58.5	0.06
C. Resource	One	56.3	46.5	46.6
	Two	*0.04*	39.5	0.09

Table 11.3: Mean precision/recall results for the collocation measures on both steps; bold number denote the maximum average for step one while italic denotes maximum average for step 2

Table 11.3 shows the results for the two first steps with these basic measures. We can see that for step one, there is a good improvement for recall by using the Tag collocation distance with respect to the Resource collocation distance. Note that getting a good recall with a clustering algorithm is not so difficult as clustering all the elements in one single cluster would give a recall of 100%, however, this would lead to a very bad precision. In the same way, if all elements are put in a distinct cluster, then the precision will be of 100% but the recall might be low.

For step one, the distances have different behaviours, but when balancing precision and recall, they reach a similar result with an F_1 measure between 47% and 57%. However, for step two, the precision is low, we believe this might be an issue with the validated dataset.

The clustering algorithm that is used for step one [EpKSX96] selects the number of clusters automatically, but to do so, it uses the parameter λ that defines the minimum distance needed to put two elements in the same cluster. Thus, when λ is close to one, most of the elements will be put in the same cluster; this will improve the recall of the algorithm but decrease the precision as there will be more noise in each cluster. Figures 11.10, 11.11 and 11.12 show the evolution of the precision and recall performance against the value of λ (the colour of the points). We can observe the usual trade off between precision and recall, and the direct correlation of this evolution with λ is quite clear: a high threshold for clustering falls at the bottom right, with a high recall but a low precision, while a low threshold achieves a better precision but a quite low recall. This happens with both measures A. (Figure 11.10) and B. (Figure 11.11), but measure C. (Figure 11.12), which has a lower

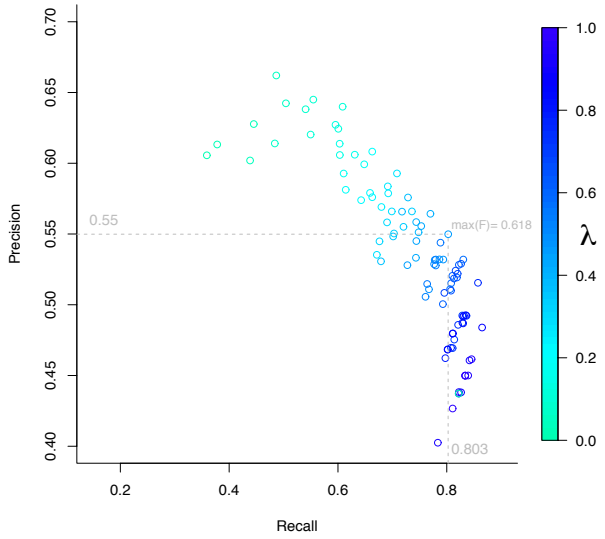


Figure 11.10: Precision/Recall Evolution of Step One with the Tag Collocation Distances; the cross marks the point with the maximum F measure: $max(F_1) = 61.8\%, P = 55\%, R = 80.3\%$

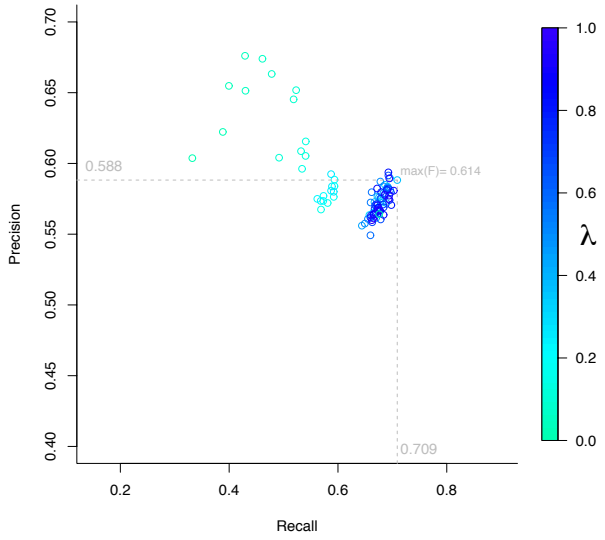


Figure 11.11: Precision/Recall Evolution of Step One with the User Collocation Distances; the cross marks the point with the maximum F measure: $max(F_1) = 61.4\%, P = 58.8\%, R = 70.9\%$

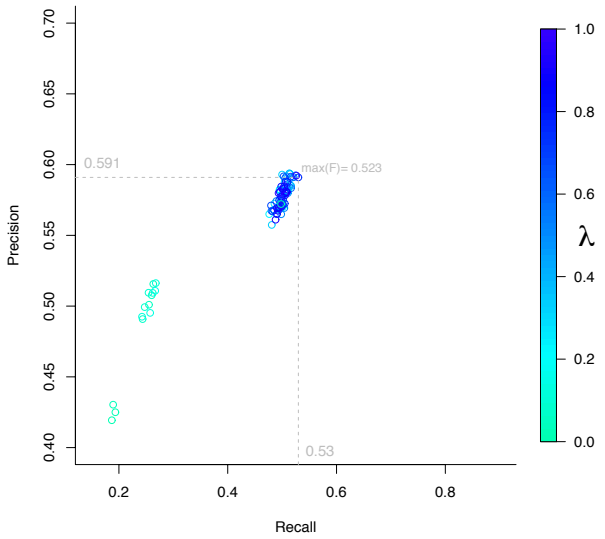


Figure 11.12: Precision/Recall Evolution of Step One with the Resource Collocation Distances; the cross marks the point with the maximum F measure: $max(F_1) = 52.3\%, P = 59.1\%, R = 53\%$

performance, behaves in a different manner as the recall increases with the precision and with λ. However, if we compare this to the two other measures, we can see that all reach a similar precision/recall pair around 60%/50%. This is the minimal trade-off when tags are clustered in a number of larger clusters, thus reaching a better recall but still keeping a separation between senses and thus keeping an average precision.

With the tag collocation measure (Figure 11.10, the results are evenly distributed, while for the user collocation and the resource collocation (Figure 11.11 and 11.12), we can see that the results are in groups at different precision/recall values within which λ does not seem to make a great difference. This might be due to the structure of the underlying folksonomy, where some distant annotations have to be put together to reach a good recall. That is, there are some annotations that are considered too distant when λ is low and are thus kept in separate clusters. λ has to significantly increase to put these distant annotations in the right clusters. However, when this happens, the recall of the clustering does get better, but distant annotations that should not be in the same cluster are also taken in and deteriorate the precision.

11.4 Summary

In this Chapter we present the evaluation of the sense induction algorithm presented in Chapter 6 "Background Knowledge Evolution". This algorithm is needed in order to increase the coverage of the linguistic and conceptual part of the background knowledge, and thus, the quality of the results of the semantic matching algorithms presented in Chapter 5.

The main contributions of this chapter are:

1. The quantification of the impact of using a static vocabulary for deriving the formal semantics in a folksonomy-based system. While the statement that the lack of background knowledge (terms, concepts and their relations) is a problem for semantic systems is present in much of the state of the art, to the best of our knowledge, there is no quantification of how much of a problem it is. In this Chapter we see that depending of the domain, the coverage of the vocabulary ranges from 49% in highly dynamic domains such as Computer Science, to 71% in more static domains such as Cooking, Travel and Education.

2. The creation of a publicly available gold standard dataset linking folksonomy annotations to Wordnet senses that can be used to evaluate sense induction, Natural Language Processing (NLP) and Word Sense Disambiguation (WSD) algorithms.

3. The proposal of an evaluation mechanism for the sense induction algorithm that considers all the three steps of the algorithm, specifically, including the novel step of being able to discover the different senses in which an annotation is used in folksonomy based systems. The proposal extends on existing evaluation mechanisms from NLP, adapting it to fit the folksonomy characteristics.

Part VI

Concluding Remarks

Chapter 12

Conclusion

In this thesis we study the issue of how to capture and manage the identity of entities in a distributed scenario by considering the real world semantics of the entities in order to enable interoperability and collaboration among the participants in a distributed scenario.

We first identify several sources of semantic heterogeneity and propose a Knowledge Organization (KOS) system architecture that deals with them by construction. The proposed KOS has three components: the *background knowledge* deals with the issue of ambiguity of the natural language by linking words with formal concepts, enabling automatic reasoning in the system; the *schematic knowledge* defines the model for the entities thus dealing with issues of interoperability at schema level; and the *concrete knowledge* defines the model for the digital entities that captures the real world semantics of entities (those that exist) as provided partially (with incomplete and inconsistent descriptions) by the mental model of entities created and maintained by people from different perspectives using different terms and languages. The digital entity solves the problems with names and descriptions of mental entities by identifying the real world entity being referred (with $SURI$s) differently from the local perspectives provided by people (identified by SURLs).

We model the distributed scenarios with three layers in order to deal with the locality and globality issues of semantics of entities. Entities are local as they are provided and maintained locally by people, and also global as there is a need of share understanding if the system is to enable interoperability and collaboration among the participants. The three layers proposed are the Universal (global), Community (glocal) and Personal (local) layers. At the personal layer people create and maintain their own view of the world modeled with entities (i.e., entities are created mostly in a bottom-up fashion) while reusing the language and concepts defined in the background knowledge from the upper layers, i.e., the social contexts where they interact with each other (their communities). However, in order to achieve multi-context interoperability there is the need of a layer above the community that maintains the knowledge that is shared among them. This mean that the language and concepts are mostly maintained in a top-down fashion.

In order to deal with the dynamic nature of knowledge, especially at the language and conceptual level, we propose a sense induction algorithm that given the usage of terms in collaborative scenarios, such as those of folksonomies, allows new concepts and terms to emerge in a bottom-up fashion. The advantage of this approach is the distribution of the workload among the participants of the system, and the fast formalization of changes in dynamic domains. The novelty of the defined sense induction algorithm is its capability for recognizing the different senses in which a word can be used in different, but also overlapping, communities. We also propose an algorithm for propagating the new knowledge defined at the different components of our KOS between the different layers.

Finally, we define a set of compatibility conditions that allows us to address the issue of multiple names and descriptions coming from multiple (incomplete and inconsistent) views of the entities being referred. The novelties of this compatibility condition are the recognition of the importance of *names* as references to the real world entities, thus allowing us to apply tailored name matching techniques on them, and the identification of a subset of the available descriptive attributes, called identifying sets, that can uniquely and globally identify the real world entities being referred.

Based on these compatibility conditions we define our *Identity Managements Framework* that is able to assign identifiers to the real world entities being referred by the participants of the distributed system, while also being able to identify and dereference the different local perspectives. The identifier of the real world entity is called $SURI$ (from semantified URI), while the identifier of the local perspective is called $SURL$ (from semantified URL). We enable interoperability be assigning both identifiers to the digital entities, therefore, regardless of the description used locally, we can know (via the $SURI$) when different local descriptions refer to the same real world entity. We define the conditions for assigning the $SURI$ given the life-cycle of the digital entities, i.e., how each create, update and delete operation affects the identity of the entity in the distributed system, so that whenever a new entity is created, updated or deleted, we can still consistently manage the $SURI$s and therefore achieve interoperability.

To summarize, **the main scientific achievements include**:

1. The definition of an architecture for managing the identity of entities in the digital domain in a distributed scenario by capturing the real world semantics of entities. The separation of entities in three types proved to be key to proposing an architecture that once and for all can solve the identifier issues in digital entities. These are:

    - real world entities, i.e., those that exist;
    - mental entities, i.e., those that are capture and provided by people from different perspectives, and

- digital entities, i.e., those that are encoded in the system capturing the semantics of real world entities as provided by people.

This separation allowed the creation of identifiers for the local perspectives (mental entities) and the globally unique identifiers for real world entities.

2. The understanding of different sources of semantic heterogeneity and the definition of a Knowledge Organization System that by design deals with them at different levels:
   - At the background knowledge level by separating the natural language from the conceptual level;
   - At the schematic level by allowing the definition of entity types based on concepts of the previous level;
   - At the concrete level by putting entities as first class citizens of the system, considering the difference between a particular perspective and the globally unique real world entity being referred.
3. The definition of tools for:
   - Semantically comparing values, by converting string to semantic strings;
   - The evolution of the background knowledge by inducing new senses based on the usage of terms by users;
   - The propagation of new knowledge between users and communities;
   - The comparison of entities based on identifying sets of attributes and names.

The contributions in terms of **implemented tools** are:

- **Structure Preserving Semantic Matching (SPSM)** inside the S-Match open source framework available at (http://s-match.org/).
- A generic sense induction algorithm that can be extended with new distance metrics.
- **Tags2Con gold standard dataset creation tool**, open source at (`http://sourceforge.net/projects/tags2con/?source=directory`) that aims at converting free-text annotations to annotations that have a machine processable meaning to it.

The contribution in terms of evaluation are:

- Structure Preserving Semantic Matching (SPSM): The evaluation of SPSM in synthetic scenarios with three types of alterations: syntactic, semantic and combined. The F-measure results (0.8; 0.73; 0.47) respectively show the robustness of the matching tool. Furthermore, in the worst performance of the tool, the F-measure is equal to base syntactic matchers such as Edit Distance.

- Considering the knowledge evolution:
  - The proposal of an evaluation mechanism for the sense induction algorithm that considers all the three steps of the algorithm, specifically, including the novel step of being able to discover the different senses in which an annotation is used in folksonomy based systems. The proposal extends on existing evaluation mechanisms from NLP, adapting it to fit the folksonomy characteristics.
  - The creation of a publicly available gold standard dataset, created using the Tags2Con tool, linking folksonomy annotations to Wordnet senses that can be used to evaluate sense induction, Natural Language Processing (NLP) and Word Sense Disambiguation (WSD) algorithms. The dataset is available at `http://disi.unitn.it/~knowdive/dataset/delicious/` and is also included in the Linked Open Data cloud as from September 2011, available at `http://richard.cyganiak.de/2007/10/lod/`.
  - The quantification of the impact of using a static vocabulary for deriving the formal semantics in a folksonomy-based system. While the statement that the lack of background knowledge (terms, concepts and their relations) is a problem for semantic systems is present in much of the state of the art, to the best of our knowledge, there is no quantification of how much of a problem it actually is. In this thesis we have seen that depending of the domain, the coverage of the vocabulary ranges from 49% in highly dynamic domains such as Computer Science, to 71% in more static domains such as Cooking, Travel and Education.

Chapter 13

Future work

The implementation and evaluation of the performance of the identity management framework, integrating all the already developed components is let as a future work. The simulation of the distributed environment and the dynamic nature of knowledge are the most challenging issues that need to captured by the evaluation of the framework. Furthermore, the availability of a dataset that contains different perspectives about the same real world entity is still to be checked. In case no such dataset can be found, we envision the creation of synthetic scenarios where the different perspectives are created based on an original entity. Some possible sources for original entities are *Yago*[1] for entities of type Person and Location, *Geonames*[2] for Locations, the *Internet Movie Database* (IMDB)[3] or the *Lumiere* database of movies released in Europe[4] for Movies, Actors, Directors and Locations, or the *US census data*[5] for Locations of the US.

The problem of name matching in a distributed scenario can be considered as a interesting research topic on its own. While there is work related to name matching, and even a book is dedicated to it, as we saw in this thesis, considering the problem in a distributed scenario where each person can define its own local names for entities, perhaps using indexicals (e,g., my mother), nicknames, translations, among others, complicates the name matching problem even more. These new issues have not been yet accounted in the literature.

While we provide an extensible algorithm for sense induction, more tests need to be performed to asses which distance measures are the best. Furthermore, while the manually created dataset is a good initial start point for the evaluation, more terms need to be validated and added to the dataset in order to provide more richness to it. This will enable the dataset to be used by more different distance metrics in the sense induction algorithm.

[1] `http://www.mpi-inf.mpg.de/yago-naga/yago/`
[2] `http://www.geonames.org/`
[3] `http://www.imdb.com/`
[4] `http://lumiere.obs.coe.int/web/sources/`
[5] `http://www.census.gov/main/www/cen1990.html`

Bibliography

[ABK+08] S. Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives, *DBpedia: A nucleus for a web of open data*, The Semantic Web, Springer, 2008, pp. 722–735.

[ACMT08] B. Alexe, L. Chiticariu, R.J. Miller, and Wang-Chiew Tan, *Muse: Mapping understanding and design by example*, Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on, april 2008, pp. 10 –19.

[AGAV09] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo, *A comparison of extrinsic clustering evaluation metrics based on formal constraints*, Information Retrieval **12** (2009), 461–486, 10.1007/s10791-008-9066-8.

[AGH07] Au, N. S. Gibbins, and N. Hadbolt, *Understanding the Semantics of Ambiguous Tags in Folksonomies*, The International Workshop on Emergent Semantics and Ontology Evolution (ESOE2007) at ISWC/ASWC 2007, November 2007.

[AM02a] Enrique Alfonseca and Suresh Manandhar, *Extending a lexical ontology by a combination of distributional semantics signatures*, Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web (London, UK), EKAW '02, Springer-Verlag, 2002, pp. 1–7.

[AM02b] Enrique Alfonseca and Suresh Manandhar, *Proposal for evaluating ontology refinement methods*, Language Resources and Evaluation (2002).

[AMO+04] 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*, Database Systems for Advances Applications (DASFAA 2004), Proceedings, Lecture Notes in Computer Science, Springer, March 2004, pp. 25–38.

[AN07] M Ames and M Naaman, *Why we tag: motivations for annotation in mobile and online media*, CHI 2007 - Tags, Tagging and Notetaking (San Jose), 2007, pp. 971–980.

[AP08] Ron Artstein and Massimo Poesio, *Inter-coder agreement for computational linguistics*, Comput. Linguist. **34** (2008), 555–596.

[APZ09] Pierre Andrews, Juan Pane, and Ilya Zaihrayeu, *Report on methods and algorithms or bootstrapping semantic web content from user repositories and reaching consensus on the use of semantics*, Tech. report, Insemtives.eu, Nov. 2009.

[APZ11a] Pierre Andrews, Juan Pane, and Ilya Zaihrayeu, *Semantic disambiguation in folksonomy: A case study*, Advanced Language Technologies for Digital Libraries (Raffaella Bernardi, Sally Chambers, Bjrn Gottfried, Frdrique Segond, and Ilya Zaihrayeu, eds.), Lecture Notes in Computer Science, vol. 6699, Springer Berlin / Heidelberg, 2011, pp. 114–134.

[APZ11b] Pierre Andrews, Juan Pane, and Ilya Zaihrayeu, *Sense induction in folksonomies*, Workshop on Discovering Meaning On the Go in Large Heterogeneous Data 2011 (Fiona McNeill, Harry Halpin, and Michael Chan, eds.), International Joint Conference on Artificial Intelligence, 2011.

[ASIBB11] Andrea Matteini Andreas Schultz, Robert Isele, Christian Bizer, and Christian Becker, *Ldif - linked data integration framework*, pp. 1–4, 2011.

[AZP+10] Pierre Andrews, Ilya Zaihrayeu, Juan Pane, Aliaksandr Autayeu, and Marin Nozhchev, *Insemtives deliverable 2.4: Report on the refinement of the proposed models, methods and semantic search*, Tech. report, Insemtives.eu, November 2010.

[AZP12] Pierre Andrews, Ilya Zaihrayeu, and Juan Pane, *A classification of semantic annotation systems*, Semantic Web Journal (2012), no. Interoperability, Usability, Applicability, to appear.

[BBR11] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm, *Schema matching and mapping*, 1st ed., Springer Publishing Company, Incorporated, 2011.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, *The description logic handbook: Theory, implementation and applications*, Cambridge University Press, 2003.

[BL94] T. Berners-Lee, *Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web*, RFC 1630 (Informational) `http://www.ietf.org/rfc/rfc1630.txt`, June 1994.

[BL09] Samuel Brody and Mirella Lapata, *Bayesian word sense induction*, Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (Stroudsburg, PA, USA), EACL '09, Association for Computational Linguistics, 2009, pp. 103–111.

[BLFM05] T. Berners-Lee, R. Fielding, and L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, RFC 3986 (Standard) `http://www.ietf.org/rfc/rfc3986.txt`, January 2005.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila, *The semantic web*, The Scientific American **284** (2001), no. 5, 28–37.

[BLMM94] T. Berners-Lee, L. Masinter, and M. McCahill, *Uniform Resource Locators (URL)*, RFC 1738 (Proposed Standard) `http://www.ietf.org/rfc/rfc1738.txt`, December 1994, Obsoleted by RFCs 4248, 4266, updated by RFCs 1808, 2368, 2396, 3986, 6196, 6270.

[BLN86] C. Batini, M. Lenzerini, and S. B. Navathe, *A comparative analysis of methodologies for database schema integration*, ACM Comput. Surv. **18** (1986), 323–364.

[BMR11] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm, *Generic schema matching, ten years later*, PVLDB **4** (2011), no. 11, 695–701.

[Bon01] A. Bonarini, *Evolutionary learning, reinforcement learning, and fuzzy rules for knowledge acquisition in agent-based systems*, Proceedings of the IEEE **89** (2001), no. 9, 1334–1346.

[Bra03] L. Karl Branting, *A comparative evaluation of name-matching algorithms*, Proceedings of the 9th international conference on Artificial intelligence and law (New York, NY, USA), ICAIL '03, ACM, 2003, pp. 224–232.

[Bra10] David Braun, *Indexicals*, The Stanford Encyclopedia of Philosophy (Edward N. Zalta, ed.), Stanford University, summer 2010 ed., 2010.

[BRIW06] Marcus Berliant, Robert R. Reed III, and Ping Wang, *Knowledge exchange, matching, and agglomeration*, Journal of Urban Economics **60** (2006), no. 1, 69–95.

[BSB08] Paolo Bouquet, Heiko Stoermer, and Barbara Bazzanella, *An entity name system (ens) for the semantic web*, Proceedings of the 5th European semantic web conference on The semantic web: research and applications (Berlin, Heidelberg), ESWC'08, Springer-Verlag, 2008, pp. 258–272.

[BSG07] Paolo Bouquet, Heiko Stoermer, and Daniel Giacomuzzi, *OKKAM: Enabling a Web of Entities*, Proceedings of the WWW2007 Workshop i3: Identity, Identifiers and Identification, Banff, Canada, May 8 2007 (Paolo Bouquet, Heiko Stoermer, Giovanni Tummarello, and Harry Halpin, eds.), CEUR Workshop Proceedings, ISSN 1613-0073, 2007.

[BSW+07] Simone Braun, Andreas Schmidt, Andreas Walter, Gabor Nagypal, and Valentin Zacharias, *Ontology maturing: a collaborative web 2.0 approach to ontology engineering*, Proceedings of (CKC 2007) at (WWW2007), 2007.

[BV11] Angela Bonifati and Yannis Velegrakis, *Schema matching and mapping: from usage to evaluation*, Proceedings of the 14th International Conference on Extending Database Technology (New York, NY, USA), EDBT/ICDT '11, ACM, 2011, pp. 527–529.

[CDMS08] Prabhakar Raghavan Christopher D. Manning and Hinrich Schtze, *Flat clustering*, Introduction to Information Retrieval, Cambridge University Press, 2008.

[CEH+08] Caterina Caracciolo, Jérôme Euzenat, Laura Hollink, Ryutaro Ichise, Antoine Isaac, Véronique Malaisé, Christian Meilicke, Juan Pane, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, and Vojtech Svátek, *Results of the ontology alignment evaluation initiative 2008*, Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008) Collocated with the 7th International Semantic Web Conference (ISWC-2008) (Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, eds.), CEUR Workshop Proceedings, vol. 431, CEUR-WS.org, 2008.

[Chr06] Peter Christen, *A comparison of personal name matching: Techniques and practical issues*, Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on, dec. 2006, pp. 290 –294.

[CSH06] Namyoun Choi, Il-Yeol Song, and Hyoil Han, *A survey on ontology mapping*, SIGMOD Record **35** (2006), no. 3, 34–41.

[DG09] B. Dutta and F. Giunchiglia, *Semantics are actually used*, International Conference on Semantic Web and Digital Libraries (Trento, Italy), 2009, pp. 62–78.

[DH05] AnHai Doan and Alon Y Halevy, *Semantic integration research in the database community: A brief survey*, AI Magazine **26** (2005), no. 1, 83.

[DJLW08] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang, *Image retrieval: Ideas, influences, and trends of the new age*, ACM Comput. Surv. **40** (2008), 5:1–5:60.

[DS04] Mike Dean and Guus Schreiber, *Owl web ontology language reference. w3c recommendation 10 february 2004*, `http://www.w3.org/TR/owl-ref/`, February 2004.

[DS05] M. Duerst and M. Suignard, *Internationalized Resource Identifiers (IRIs)*, RFC 3987 (Standard) `http://www.ietf.org/rfc/rfc3987.txt`, January 2005.

[DvGIF99] L. Daigle, D. van Gulik, R. Iannella, and P. Falstrom, *URN Namespace Definition Mechanisms*, RFC 2611 (Best Current Practice) `http://www.ietf.org/rfc/rfc2611.txt`, June 1999, Obsoleted by RFC 3406.

[EFH+09] Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, François Scharffe, Pavel Shvaiko, Vassilis Spiliopoulos, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, Cássia Trojahn dos Santos, George A. Vouros, and Shenghui Wang, *Results of the ontology alignment evaluation initiative 2009*, in Shvaiko et al. [SEG+09].

[EFM+10] Jérôme Euzenat, Alfio Ferrara, Christian Meilicke, Juan Pane, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, and Cássia Trojahn dos Santos, *Results of the ontology alignment evaluation initiative 2010*, Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) (Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Ming Mao, and Isabel F. Cruz, eds.), CEUR Workshop Proceedings, vol. 689, CEUR-WS.org, 2010.

[EIM+07] Jérôme Euzenat, Antoine Isaac, Christian Meilicke, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb, Vojtech Svátek,

Willem Robert van Hage, and Mikalai Yatskevich, *Results of the ontology alignment evaluation initiative 2007*, in Shvaiko et al. [SEGH08].

[EpKSX96] Martin Ester, Hans peter Kriegel, Jrg S, and Xiaowei Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), AAAI Press, 1996, pp. 226–231.

[ES07] Jérôme Euzenat and Pavel Shvaiko, *Ontology matching*, Springer-Verlag New York, Inc., 2007.

[Fel98] Christiane Fellbaum (ed.), *Wordnet an electronic lexical database*, The MIT Press, Cambridge, MA ; London, May 1998.

[FFMM94] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire, *KQML as an agent communication language*, Proceedings of the third international conference on Information and knowledge management (New York, NY, USA), CIKM '94, ACM, 1994, p. 456463.

[FGT11] Martin Fenner, Consol Garcia Gmez, and Gudmundur A. Thorisson, *Collective action for the open researcher and contributor id (orcid)*, Serials: The Journal for the Serials Community **24** (2011), 277–279.

[FKEV07] Jean Francois, Djoufak Kengue, Jérôme Euzenat, and Petko Valtchev, *OLA in the OAEI 2007 evaluation contest*, ISWC (2007), CiteSeerX, 2007.

[FMK+08] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou, *Ontology change: Classification and survey*, Knowl. Eng. Rev. **23** (2008), no. 2, 117–152.

[Ful94] Rodney Fuller, *How computers affect interpersonal communications*, The Arachnet Electronic Journal on Virtual Culture **2** (1994), no. 2.

[GAP12] Fausto Giunchiglia, Aliaksandr Autayeu, and Juan Pane, *S-Match: an open source framework for matching lightweight ontologies*, Semantic Web Journal (2012), no. Semantic Web Tools and Systems, to appear.

[GD11] Fausto Giunchiglia and Biswanath Dutta, *Dera: A faceted knowledge organization framework*, Tech. Report DISI-11-457, Ingegneria e Scienza dell'Informazione, University of Trento. Italy, 2011.

[GDMF11] Fausto Giunchiglia, Biswanath Dutta, Vincenzo Maltese, and Feroz Farazi, *A facet-based methodology for the construction of a large-scale geo-spatial ontology*, Tech. Report DISI-11-479, Ingegneria e Scienza dell'Informazione, University of Trento. Italy, 2011.

[GH06] Scott Golder and Bernardo A. Huberman, *The structure of collaborative tagging systems*, Journal of Information Science **32** (2006), no. 2, 198–208.

[GM10] Fausto Giunchiglia and Vincenzo Maltese, *Ontologie leggere a faccette*, Tech. Report DISI-10-005, Ingegneria e Scienza dell'Informazione, University of Trento. Italy, 2010.

[GMA09] Fausto Giunchiglia, Vincenzo Maltese, and Aliaksandr Autayeu, *Computing minimal mappings*, Proceedings of the 4th Workshop on Ontology matching at ISWC, 2009.

[GMD12] Fausto Giunchiglia, Vincenzo Maltese, and Biswanath Dutta, *Domains and context: first steps towards managing diversity in knowledge*, Web Semantics: Science, Services and Agents on the World Wide Web **12** (2012), no. 0.

[GMFD10] Fausto Giunchiglia, Vincenzo Maltese, Feroz Farazi, and Biswanath Dutta, *Geowordnet: A resource for geo-spatial applications*, The Semantic Web: Research and Applications (Lora Aroyo, Grigoris Antoniou, Eero Hyvnen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, eds.), Lecture Notes in Computer Science, vol. 6088, Springer Berlin / Heidelberg, 2010, pp. 121–136.

[GMY+08] Fausto Giunchiglia, Fiona McNeill, Mikalai Yatskevich, Juan Pane, Paolo Besana, and Pavel Shvaiko, *Approximate structure-preserving semantic matching*, Proceedings of the OTM 2008 Confederated International Conferences. ODBASE 2008 (Moterrey, Mexico), Springer-Verlag, 2008, pp. 1217–1234.

[GMZ07] Fausto Giunchiglia, Maurizio Marchese, and Ilya Zaihrayeu, *Encoding classifications into lightweight ontologies*, Journal on Data Semantics VIII, 2007, pp. 57–81.

[Gro04] Web Ontology Working Group, *W3c recommendation for owl*, `http://www.w3.org/TR/2004/REC-owl-features-20040210/`, February 2004.

[Gru93] Thomas R. Gruber, *A translation approach to portable ontology specifications*, Knowledge Acquisition **5** (1993), no. 2, 199–220.

[GS03] Fausto Giunchiglia and Pavel Shvaiko, *Semantic matching*, The Knowledge Engineering Review **18** (2003), no. 3, 265–280.

[GSCAGP10] A. García-Silva, O. Corcho, H. Alani, and A. Gómez-Perez, *Review of the state of the art: Discovering and associating semantics to tags in folksonomies*, The Knowledge Engineering Review (2010).

[GSSAC09] A. García-Silva, M. Szomszor, H. Alani, and O. Corcho, *Preliminary results in tag disambiguation using dbpedia*, Proc. of KCAP (USA), 2009.

[GW92] Fausto Giunchiglia and Toby Walsh, *A theory of abstraction*, Artificial Intelligence **57** (1992), no. 23, 323 – 389.

[GYAS09] Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, and Pavel Shvaiko, *A large scale dataset for the evaluation of ontology matching systems*, The Knowledge Engineering Review Journal **24** (2009), no. 2, 137–157.

[GYS07] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko, *Semantic matching: Algorithms and implementation*, Journal on Data Semantics IX, 2007, pp. 1–38.

[Hal05] Alon Halevy, *Why your data won't mix*, Queue **3** (2005), 50–58.

[HD04] Geoff Holloway and Mike Dunkerley, *The math, myth and magic of name search and matching*, 5th ed., SearchSoftwareAmerica, 2004.

[HHD07] Aidan Hogan, Andreas Harth, and Stefan Decker, *Performing object consolidation on the semantic web data graph*, Proceedings of I3: Identity, Identifiers, Identification, in conjunction with 16th International World Wide Web Conference (WWW 2007) (2007).

[HHSS05] Peter Haase, Andreas Hotho, Lars Schmidt-Thieme, and York Sure, *Collaborative and Usage-Driven evolution of personal ontologies*, The Semantic Web: Research and Applications, 2005, pp. 486–499.

[HM93] Joachim Hammer and Dennis Mcleod, *An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems*, International Journal of Intelligent and Cooperative Information Systems **2** (1993), no. 1, 51–83.

[Hod00] Gail Hodge, *Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files*, Council on Library and Information Resources, April 2000.

[HS98] Udo Hahn and Klemens Schnattinger, *Towards text knowledge engineering*, IN AAAI/IAAI, 1998, pp. 524–531.

[HZU+12] Aidan Hogan, Antoine Zimmermann, Juergen Umbrich, Axel Polleres, and Stefan Decker, *Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora*, Web Semantics: Science, Services and Agents on the World Wide Web **10** (2012), no. 0.

[IJB10] Robert Isele, Anja Jentzsch, and Christian Bizer, *Silk server - adding missing links while consuming linked data*, 2010.

[IV98] Nancy Ide and Jean Véronis, *Introduction to the special issue on word sense disambiguation: the state of the art*, Comput. Linguist. **24** (1998), 2–40.

[Jam09] Salma Jamoussi, *Une nouvelle representation vectorielle pour la classification semantique*, Apprentissage automatique por le TAL, vol. 50, TAL, 2009, pp. 23–57.

[JGM08] Afraz Jaffri, Hugh Glaser, and Ian Millard, *Managing URI synonymity to enable consistent reference on the semantic web*, IRSW2008 - Identity and Reference on the Semantic Web 2008 (2008).

[JMSK09] Yves R. Jean-Mary, E. Patrick Shironoshita, and Mansur R. Kabuka, *Ontology matching with semantic verification*, Web Semantics: Science, Services and Agents on the World Wide Web **7** (2009), no. 3, 235 – 251, ¡ce:title¿The Web of Data¡/ce:title¿.

[KEV07] Jean François Djoufak Kengue, Jérôme Euzenat, and Petko Valtchev, *Ola in the oaei 2007 evaluation contest*, in Shvaiko et al. [SEGH08].

[KS03] Yannis Kalfoglou and Marco Schorlemmer, *Ontology mapping: The state of the art*, The Knowledge Engineering Review **18** (2003), no. 01, 1–31.

[KS10] Christian Körner and Markus Strohmaier, *A call for social tagging datasets*, SIGWEB Newsl. (2010), 2:1–2:6.

[LDZ09] Huairen Lin, Joseph Davis, and Ying Zhou, *An integrated approach to extracting ontological structures from folksonomies*, The Semantic Web: Research and Applications (Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvnen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, eds.), Lecture Notes in Computer Science, vol. 5554, Springer, Berlin / Heidelberg, 2009, pp. 654–668.

[Lin98] Dekang Lin, *Automatic retrieval and clustering of similar words*, Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2 (Stroudsburg, PA, USA), ACL '98, Association for Computational Linguistics, 1998, pp. 768–774.

[LS99] Ora Lassila and Ralph R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification*, Tech. Report REC-rdf-syntax-19990222, World Wide Web Consortium (W3C), Cambridge, MA, 1999.

[LSDR07] Yoonkyong Lee, Mayssam Sayyadian, AnHai Doan, and Arnon Rosenthal, *etuner: tuning schema matching software using synthetic scenarios*, The VLDB Journal **16** (2007), 97–122, 10.1007/s00778-006-0024-z.

[Lud11] Peter Ludlow, *Descriptions*, The Stanford Encyclopedia of Philosophy (Edward N. Zalta, ed.), winter 2011 ed., 2011.

[MBG+02] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, Alessandro Oltramari, Ro Oltramari, Luc Schneider, Lead Partner Istc-cnr, and Ian Horrocks, *Wonderweb deliverable d17. the wonderweb library of foundational ontologies and the dolce ontology*, 2002.

[MBPG09] Fiona McNeill, Paolo Besana, Juan Pane, and Fausto Giunchiglia, *Service integration through structure preserving semantic matching*, Cases on Semantic Interoperability for Information Systems Integration:Practices and Applications **11** (2009), no. Semantic Interoperability, 26–45.

[MDA08] Mohamed Zied Maala, Alexandre Delteil, and Ahmed Azough, *A conversion process from Flickr tags to RDF descriptions*, IADIS international journal on www/internet **6** (2008), no. 1 (en).

[Mik05] Peter Mika, *Ontologies are us: A unified model of social networks and semantics*, International Semantic Web Conference, 2005, pp. 522–536.

[ML11] Eric Margolis and Stephen Laurence, *Concepts*, The Stanford Encyclopedia of Philosophy (Edward N. Zalta, ed.), Stanford University, fall 2011 ed., 2011.

[MSD08] R. McCann, Warren Shen, and AnHai Doan, *Matching schemas in online communities: A web 2.0 approach*, Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on, april 2008, pp. 110 –119.

[MvH04] Deborah L. McGuinness and Frank van Harmelen, *OWL Web Ontology Language Overview*, Tech. report, World Wide Web Consortium (W3C), February 2004.

[OAH+07] Daniel Oberle, Anupriya Ankolekar, Pascal Hitzler, Philipp Cimiano, Michael Sintek, Malte Kiesel, Babak Mougouie, Stephan Baumann, Shankar Vembu, Massimo Romanelli, Paul Buitelaar, Ralf Engel, Daniel Sonntag, Norbert Reithinger, Berenike Loos, Hans-Peter Zorn, Vanessa Micelli, Robert Porzel, Christian Schmidt, Moritz Weiten, Felix Burkhardt, and Jianshen Zhou, *Dolce ergo sumo: On foundational and domain models in the smartweb integrated ontology (swinto)*, Web Semantics: Science, Services and Agents on the World Wide Web **5** (2007), no. 3, 156 – 174.

[PH00] Charnyote Pluempitiwiriyawej and Joachim Hammer, *A classification scheme for semantic and schematic heterogeneities in xml data sources*, Tech. report, University of Florida, Gainesville, FL, September 2000.

[PNL02] Adam Pease, Ian Niles, and John Li, *The suggested upper merged ontology: A large ontology for the semantic web and its applications*, In Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web, 2002.

[RMT08] Francesco Ronzano, Andrea Marchetti, and Maurizio Tesconi, *Tagpedia: a semantic reference to describe and search for web resources*, SWKM 2008: Intl. Workshop on Social Web and Knowledge Management @ WWW, 2008.

[Rus05] Bertrand Russell, *On denoting*, Mind **14** (1905), no. 56, pp. 479–493 (English).

[SC08] Leo Sauermann and Richard Cyganiak, *Cool uris for the semantic web. w3c interest group note 03*, `http://www.w3.org/TR/cooluris/`, December 2008.

[Sch06] P. Schmitz, *Inducing ontology from flickr tags*, Proc. of the Collaborative Web Tagging Workshop (WWW 06), May 2006.

[Sch07] S.E. Schaeffer, *Graph clustering*, Computer Science Review **1** (2007), no. 1, 27–64.

[SE05] Pavel Shvaiko and Jérôme Euzenat, *A survey of schema-based matching approaches*, Journal on Data Semantics IV (Stefano Spaccapietra, ed.), Lecture Notes in Computer Science, vol. 3730, Springer Berlin / Heidelberg, 2005, pp. 146–171.

[Sea58] John R. Searle, *Proper names*, Mind **67** (1958), no. 266, pp. 166–173 (English).

[SEG+09] Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Natalya Fridman Noy, and Arnon Rosenthal (eds.), *Proceedings of the 4th international workshop on ontology matching (om-2009) collocated with the 8th international semantic web conference (iswc-2009) chantilly, usa, october 25, 2009*, CEUR Workshop Proceedings, vol. 551, CEUR-WS.org, 2009.

[SEGH08] Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Bin He (eds.), *Proceedings of the 2nd international workshop on ontology matching (om-2007) collocated with the 6th international semantic web conference (iswc-2007) and the 2nd asian semantic web conference (aswc-2007), busan, korea, november 11, 2007*, CEUR Workshop Proceedings, vol. 304, CEUR-WS.org, 2008.

[SK05] Marco Schorlemmer and Yannis Kalfoglou, *Progressive ontology alignment for meaning coordination: an information-theoretic foundation*, Knowledge Creation Diffusion Utilization (2005), 737–744.

[SK08] Marco Schorlemmer and Yannis Kalfoglou, *Institutionalising ontology-based semantic integration*, Appl. Ontol. **3** (2008), 131–150.

[SM07] L. Specia and E. Motta, *Integrating folksonomies with the semantic web*, Proc. of the European Semantic Web Conference (ESWC2007) (Berlin Heidelberg, Germany), LNCS, vol. 4519, Springer-Verlag, July 2007, pp. 624–639.

[SMMS02] Ljiljana Stojanovic, Alexander Maedche, Boris Motik, and Nenad Stojanovic, *User-Driven ontology evolution management*, Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, 2002, pp. 133–140.

[SR09] Heiko Stoermer and Nataliya Rassadko, *Results of okkam feature based entity matching algorithm for instance matching contest of oaei 2009*, in Shvaiko et al. [SEG+09].

[SRT05] Amit Sheth, Cartic Ramakrishnan, and Christopher Thomas, *Semantics for the semantic web: the implicit, the formal and the powerful*, International Journal on Semantic Web and Information Systems **1** (2005), 1–18.

[Ste02] Mark Stevenson, *Word sense disambiguation*, CSLI Publications, Stanford, CA, USA, 2002.

[Tai79] Kuo-Chung Tai, *The tree-to-tree correction problem*, J. ACM **26** (1979), 422–433.

[UG04] M. Uschold and M. Gruninger, *Ontologies and semantics for seamless connectivity*, SIGMOD Rec. **33** (2004), no. 4, 58–64.

[vA06] Luis von Ahn, *Games with a Purpose*, IEEE Computer **39** (2006), no. 6, 92–94.

[vAMM+08] Luis von Ahn, Benjamin Maurer, Colin Mcmillen, David Abraham, and Manuel Blum, *reCAPTCHA: Human-Based Character Recognition via Web Security Measures*, Science (2008), 1160379+.

[VHS07] Caline Van Damme, Martin Hepp, and Katharina Siorpaes, *Folksontology: An integrated approach for turning folksonomies into ontologies*, Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007), 2007, pp. 57–70.

[VSP+12] Lorenzino Vaccari, Pavel Shvaiko, Juan Pane, Paolo Besana, and Maurizio Marchese, *An evaluation of ontology matching in geo-service applications*, GeoInformatica (Journal) **16** (2012), 31–66.

[Wal07] Thomas Vander Wal, *Folksonomy: Coinage and definition*, http://www.vanderwal.net/folksonomy.html, 2007.

[WSVZ08] Kilian Quirin Weinberger, Malcolm Slaney, and Roelof Van Zwol, *Resolving tag ambiguity*, Proceeding of the 16th ACM international conference on Multimedia (New York, NY, USA), MM '08, ACM, 2008, doi:10.1145/1459359.1459375, pp. 111–120.

[WSyVZ08] Kilian Quirin Weinberger, Malcolm Slane y, and Roelof Van Zwol, *Resolving tag ambiguity*, Proceeding of the 16th ACM international conference on Multimedia (New York, NY, USA), MM '08, ACM, 2008, pp. 111–120.

[WZB08] Robert Wetzker, Carsten Zimmermann, and Christian Bauckhage, *Analyzing Social Bookmarking Systems: A del.icio.us Cookbook*, Proceedings of the ECAI 2008 Mining Social Data Workshop, IOS Press, 2008, pp. 26–30.

[XW05] Rui Xu and Donald Wunsch, *Survey of clustering algorithms*, IEEE Transactions on Neural Networks / a Publication of the IEEE Neural Networks Council **16** (2005), no. 3, 645–678, PMID: 15940994.

[YYSL08] Conglei Yao, Yongjian Yu, Sicong Shou, and Xiaoming Li, *Towards a global schema for web entities*, Proceeding of the 17th international conference on World Wide Web (New York, NY, USA), WWW '08, ACM, 2008, pp. 999–1008.

[Zho07] Lina Zhou, *Ontology learning: state of the art and open issues*, Information Technology and Management **8** (2007), no. 3, 241–252.

[ZS06] Anna Zhdanova and Pavel Shvaiko, *Community-driven ontology matching*, The Semantic Web: Research and Applications (York Sure and John Domingue, eds.), Lecture Notes in Computer Science, vol. 4011, Springer Berlin / Heidelberg, 2006, pp. 34–49.

[ZSG+07] Ilya Zaihrayeu, Lei Sun, Fausto Giunchiglia, Wei Pan, Qi Ju, Mingmin Chi, and Xuanjing Huang, *From web directories to ontologies: Natural language processing challenges*, ISWC/ASWC, 2007, pp. 623–636.

[ZWY06] Lei Zhang, Xian Wu, and Yong Yu, *Emergent Semantics from Folksonomies: A Quantitative Study*, Journal on Data Semantics VI, Lecture Notes in Computer Science, Springer, 2006, pp. 168–186.